НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

«На правах рукопису»

УДК 004.056

«До захисту допущено»
В.о. завідувача кафедри
_____ М.В.Грайворонський
“___”_____2018 р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності: 125 Кібербезпека

на тему: Метод відновлення ланцюгів подій в інформаційній системі на основі теорії графів

Виконав (-ла): студент (-ка) 2 курсу, групи ФБ-71мп
(шифр групи)

Коршун Артем Сергійович
(прізвище, ім’я, по батькові)

Науковий керівник к.т.н., доц. Барановський Олексій Миколайович _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (науковий ступінь, вчене звання, , прізвище, ініціали) (підпис)

Рецензент к.т.н., ст. викл. ОНАЗ Височіненко М.С. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою
Спеціальність (спеціалізація) – 125 Кібербезпека («Системи і технології кібербезпеки)

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ М.В.Грайворонський
(підпис)

«___»_____2018 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Коршуну Артему Сергійовичу

1. Тема дисертації: Метод відновлення ланцюгів подій в інформаційній системі на основі теорії графів

науковий керівник дисертації к.т.н., доц. Барановський Олексій Миколайович,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «15» листопада 2018 р. № 4171-с

2. Термін подання студентом дисертації 12.12.2018 р.

3. Об'єкт дослідження _____
_____

4. Вихідні дані _____
_____
_____
_____

5. Перелік завдань, які потрібно розробити _____
_____
_____
_____
_____
_____

6. Орієнтовний перелік ілюстративного матеріалу _____
_____
_____

7. Орієнтовний перелік публікацій _____

_____

_____

8. Консультанти розділів дисертації*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка

Студент _____ (підпис) _____ (ініціали, прізвище)

Науковий керівник дисертації _____ (підпис) _____ (ініціали, прізвище)

* Консультантом не може бути зазначено наукового керівника магістерської дисертації.

РЕФЕРАТ

Магістерська дисертація складається з обкладинки, титульного аркушу, завдання на магістерську дисертацію, реферату, змісту, перелік умовних позначень, символів, одиниць, скорочень і термінів, вступу, чотирьох розділів, висновків, переліку джерел посилання з 5 найменувань, 1 додатку і містить 24 малюнків, 29 таблиць. Повний обсяг магістерської дисертації становить 115 сторінок, додатки - 6 сторінок.

Об'єктом дослідження є дані які можна використовувати як докази в комп'ютерній криміналістиці.

Предметом дослідження є пошук взаємозв'язків між зібраними волатильними та не волатильними даними для відновлення ланцюгів подій.

Метою даної кваліфікаційної роботи є розробка інструментарію для відновлення ланцюгів подій в інформаційній системі.

Методами дослідження дипломної роботи є аналіз та порівняння існуючих програмних додатків для комп'ютерної криміналістики, визначення їх недолікі, та використання методів теорії графів для розробка власного методу відновлення ланцюгів подій в інформаційній системі.

Результатом дипломної роботи є власний метод для відновлення ланцюгів подій в інформаційній системі та програмний продукт який можна впроваджувати для криміналістичного розслідування.

Ключові слова. ВІДНОВЛЕННЯ ЛАНЦЮГІВ ПОДІЙ, ТЕОРІЯ ГРАФА, ФОРЕНЗІКА, ВІДНОВЛЕННЯ ПОДІЙ В ІНФОРМАЦІЙНІЙ СИСТЕМІ, ЗБІР ВОЛАТИЛЬНИХ ДАНИХ, ЗБІР НЕ ВОЛАТИЛЬНИХ ДАНИХ, ВЗАЄМОЗАЛЕЖНІСТЬ ДАНИХ, C#, WINDOWS FORM.

ABSTRACT

The master's dissertation consists of the cover, the title page, the task for the master's dissertation, the abstract, the content, the list of symbols, symbols, units, abbreviations and terms, the introduction, four sections, conclusions, a list of sources of reference from 5 titles, 1 annex and contains 24 images, 29 tables. The work includes 115 pages, applications - 6 pages.

The object of the study is data that can be used as evidence in computer forensics.

The subject of the study is the search for the relationship between the volatile and volatile data collected to restore the chain of events.

The purpose of this qualification work is to develop tools for the restoration of the chain of events in the information system.

The methods of studying the thesis are to analyze and compare existing software applications for computer criminology, to identify their disadvantages, and to use the graph theory techniques to develop their own method for restoring events in the information system.

The result of the thesis is its own method for restoring the chain of events in the information system and the software product that can be implemented for forensic investigations.

Keywords. RESTORATION OF EVENTS, THEORY OF GRAPH, FORENSION, RESTORATION OF EVENTS IN THE INFORMATION SYSTEM, COLLECTION OF VOLATILE DATA, COLLECTION OF NON VOLATILE DATA, RELATED DATA RISK, C #, WINDOWS FORM.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ACID - Atomicity, Consistency, Isolation, Durability

API - application programming interface

CMD - command line interpreter

CSS - cascading style sheets

CSV- comma-separated values

DFF - digital forensics framework

DRAM - dynamic random access memory

EXIF - Exchangeable Image File Format

FAT - file allocation table

FTP - file transfer protocol

IDE - integrated development environment

IP- Internet Protocol

NTFS - new technology file system

SDK - software development kit

SRAM - static random access memory

SSD – solid-state drive

URL - uniform resource locator

XML - extensible markup language

БД - База данных

ОЗП - опервтивно запамятовуючій пристрій

ОС - операційна система

ПЗ - програмне забезпечення

ПК - персональний комп'ютер

ЦП - центральний процесор

ВСТУП

Безпека даних є важливою у наш час. Тримання даних в електронному вигляді є дуже зручним, дані особливо великих розмірів не займають багато місця, їх легко копіювати та розповсюджувати. Але такі дані є вразливими до кібернападів. Є багато варіантів захистити систему: робити бекапи системи, моніторинг дій в інформаційній системі та за потреби перешкодити зловмиснику, встановлювати лише перевірені програмні додатки, вставлювати антивірусні програми.

Але навіть такі заходи безпеки не завжди можуть захистити систему, адже з кожним днем зловмисники знаходять нові методи та технології які використовують під час своїх кібернападів. Якщо кібернапад вже стався, то потрібно визначити якими методами та за рахунок якої вразливості відбувся напад, щоб в подальшому вжити заходів для усунення вразливості. Саме цим займається комп'ютерна криміналістика. Відновленням ланцюгів подій в інформаційній системі грає ключову роль в ній.

Об'єктом дослідження є дані які можна використовувати як докази в комп'ютерній криміналістиці.

Предметом дослідження є пошук взаємозв'язків між зібраними волатильними та не волатильними даними для відновлення ланцюгів подій.

Метою роботи є розробка інструментарію для відновлення ланцюгів подій в інформаційній системі.

Завданням дослідження є дослідити сучасні програмні засоби для комп'ютерної криміналістики та виявити недоліки в їх роботі, розробити метод для відновлення ланцюгів подій в інформаційній системі та реалізувати візуалізовану програму використовуючи цей метод.

Методами дослідження дипломної роботи є аналіз існуючих проблем в комп'ютерній криміналістиці та використання методів теорії графів для розробка власного методу відновлення ланцюгів подій в інформаційній системі.

Наукова новизна даної роботи полягає у розробці власного методу пошуку взаємозв'язків між даними не залежно від їх типу. Це э актуальним адже готові програмні рішення які є на даний момент не можуть оброблювати данні в цілому. Вони оброблюють окремо певні групи події в інформаційній системі і не пов'язують отримані результати між різнотипними групами подій. За рахунок цього користувач не може відновити послідовність подій в цілому.

Практичне значення в результаті роботи буде отримано власний метод для відновлення ланцюгів подій в інформаційній системі та програмний продукт який можна впроваджувати для криміналістичного розслідування для відновлення послідовності поді та пошуку взаємопов'язаних даних після кіберзлочену.

1 ОГЛЯД СТАНУ СУЧАСНОЇ КОМП'ЮТЕРНОЇ КРИМІНАЛІСТИКИ

В цьому розділі буде розглянуто сучасні проблеми комп'ютерної криміналістики, та описано принцип відновлення подій в інформаційній системі. Також буде розглянуто існуючі програмні продукти для відновлення подій в інформаційній системі та наведені їх недоліки.

З розвитком інформаційно-комунікаційних технологій та розширенням доступу до інтернету, організації стають вразливими до різних видів загроз. Фактично, їх інформація э вразливою до кібернападів. Загрози надходять з різних джерел, таких як діяльність працівників чи хакерські атаки.

Вразливості складаються з недоліків у системі, яка може експлуатуватися зловмисниками, які можуть призвести до небезпечного впливу на систему в цілому. Коли в системі існує вразливість, загроза може проявлятися через агент загрози, який використовує, а за допомогою особливої техніки проникнення зловмисник може отримати доступ до даних та скористатися ними. Відповідно до 11-го щорічного обстеження комп'ютерних злочинів та безпеки, 74,3% від загальної суми збитків викликане: вірусами, несанкціонованим доступом, крадіжкою ноутбуків чи мобільних пристроїв та крадіжкою власної інформації.

Дійсно, дослідження, проведене МакКью, вказує на те, що 70% випадків шахрайства здійснюють інсайдери, а не зовнішні злочинці, але 90% контролю безпеки зосереджені на зовнішніх загрозах.

Щоб забезпечити безпеку системи, потрібно вчасно виявити загрози та усунути вразливість. Якщо проникнення вже відбулося не менш можливим є встановлення причини та методу завдяки якому вдалося це зробити, що б в подальшому усунити цю вразливість тим самим захистити систему. Відстежити які файли було викрадено змінено, видалено або замінено. Саме це є основною задачею комп'ютерної криміналістики. Тому цим я и буду займатися у своїй дипломній роботі.

1.1 Сучасні проблеми комп'ютерної криміналістики

Комп'ютерна криміналістика - підрозділ криміналістики, прикладна наука про розслідування злочинів та збір цифрових доказів, що знаходяться на комп'ютерах, системах зберігання даних, в комп'ютерних мережах, на мобільних та інших цифрових пристроях.

Основні проблеми під час відновлення інформації в комп'ютерній криміналістиці:

- Відновлення «закритих» даних
- Відновлення з різних систем зберігання даних
- Відновлення зруйнованих даних

Відновлення «закритих» даних - це відновлення даних які закриті паролем, зашифровані або знаходяться в прихованих областях носія інформації.

Відновлення з різних систем зберігання даних - це відновлення даних які знаходяться RAID системи і зовнішні DAS системи, в мережевих сховищах NAS або у віртуальних і розподілених сховищах даних.

Відновлення зруйнованих даних – це відновлення даних які пошкоджені апаратним або програмним збоєм, впливом зловмисного програмного забезпечення, помилками і навмисними діями користувачів.

Зловмисне програмне забезпечення на основі методу інфікування є наступним [7]:

1. Virus - Вони мають можливість повторювати самі себе, підключивши їх до програми на головному комп'ютері, як пісні, відео тощо, а потім вони подорожуються по всьому інтернету. Ther Creeper Virus був вперше виявлений на ARPANET. Приклади включають файловий вірус, вірус макросів, вірус завантажувального сектора, вірус стелс тощо.

2. Worms - хробаки також самостійно відтворюються в системі, але вони не підключаються до програми на головному комп'ютері. Найбільша різниця між вірусом та хробаками полягає в тому, що хробаки знають про мережу. Вони можуть легко пересуватися з одного комп'ютера на інший, якщо мережа доступна,

і на цільовій машині вони не завдадуть значної шкоди, вони, наприклад, споживають місце на жорсткому диску, тим самим сповільнюючи роботу комп'ютера.

3.Trojan - їх мета полягає в тому, щоб приховати себе всередині програмного забезпечення, яке є законним і коли це програмне забезпечення виконується, вони виконають своє завдання - або крадіжку інформації, або будь-яку іншу ціль, для якої вони призначені.

4. Bots - можна розглядати як вдосконалення для хробаків. Це автоматичні процеси, призначені для взаємодії через Інтернет без необхідності взаємодії з людьми. Вони можуть бути хорошими або поганими. Шкідливий бот може заразити одного хоста і після зараження створить з'єднання з центральним сервером, який забезпечить команди всім зараженим хостам, підключеним до цієї мережі, називається Botnet.

Основні дії шкідливих програм:

1. Adware – бувають як шкідливими так і ні, але вони порушують конфіденційність користувачів. Вони розміщують рекламу на робочому столі комп'ютера або в окремих програмах. Вони приєднуються з вільним використанням програмного забезпечення, що є основним джерелом доходу для таких розробників. Вони відстежують ваші інтереси та показують релевантні об'яви. Зловмисник може вставляти шкідливий код всередині програмного забезпечення, а рекламне ПЗ може відстежувати ваші дії системи і навіть може скомпрометувати вашу машину.

2. Spyware - це програма, або ми можемо сказати програмне забезпечення, яке стежить за вашою діяльністю на комп'ютері та показує зібрану інформацію зацікавленій стороні. Шпигунські програми, як правило, випадає троянами, вірусами чи хробаками. Після того, як вони випали, вони встановлюють себе і сидять тихо, щоб уникнути виявлення.

Одним з найпоширеніших прикладів програм-шпигунів є KEYLOGGER. Основна робота кейлоггера полягає в тому, щоб записати користувальницькі

натискання клавіш з міткою часу. Таким чином, зберігається цікава інформація, така як ім'я користувача, паролі, дані кредитної картки тощо.

3. Ransomware - це тип шкідливого програмного забезпечення, який буде або шифрувати ваші файли, або заблокувати ваш комп'ютер, роблячи його недоступним частково або цілком. Потім з'явиться екран із запитом грошей, тобто викупом в обмін.

4. Rootkits - призначені для отримання адміністративних прав в системі користувача. Після того, як він отримав доступ, експлуататор може зробити що-небудь, від крадіжки приватних файлів до видалення їх.

5. Zombies - вони працюють схожий на Spyware. Інфекційний механізм однаковий, але вони не шпигують і не крадуть інформацію, а лише чекають хакерських команд.

1.2 Завдання відновлення подій в комп'ютерній криміналістиці

Метою комп'ютерної криміналістики є дослідження цифрових засобів інформації з метою виявлення, збереження, відновлення, аналізу та подання фактів та припущень щодо цифрових даних [15].

Незважаючи на те, що це найчастіше пов'язане з розслідуванням широкого спектру комп'ютерних злочинів, комп'ютерна криміналістична практика також може використовуватися в цивільному судочинстві. Дисципліна передбачає аналогічні методи та принципи відновлення даних, але з додатковими інструкціями та практикою, призначеними для створення юридичної перевірки.

Під час вивчення даних, як правило, слідують за стандартним набором процедур: після фізичного ізоляції відповідного пристрою, щоб переконатися, що він не може бути випадково забруднений, слідчі створюють цифрову копію носія інформації про пристрій. Після того як оригінальний носій був скопійований, він заблокований в безпечному або іншому безпечному об'єкті, щоб зберегти його незайманий стан. Все дослідження проводиться на цифровій копії [1].

Слідчі використовують різноманітні технічні засоби та програмне забезпечення для криміналістичних програм для перевірки копії, пошуку прихованих папок і нерозподіленого місця на диску для копій видалених, зашифрованих або пошкоджених файлів.

Злочини, пов'язані з комп'ютером, можуть відбуватися по всьому світу та спектр злочинної діяльності дуже великий, від дитячої порнографії до крадіжки особистих даних, знищення інтелектуальної власності. По-друге, експерт повинен вибрати відповідний інструменти для використання. Експерт повинен бути знайомий з безліччю методів і програмного забезпечення, щоб запобігти подальшому пошкодженню в процес відновлення. Основні техніки комп'ютерної криміналістики:

Аналіз крос-дисків - одна з технік комп'ютерної криміналістики, яка корелює інформацію, знайдену на кількох жорстких дисках. Процес, який все ще досліджується, може бути використаний для ідентифікації соціальних мереж та для виявлення аномалій

Live аналіз - дослідження комп'ютерів з операційної системи з використанням спеціальної криміналістики або існуючих інструментів для отримання доказів. Ця практика є корисною для роботи з файловими системами шифрування, наприклад, де можуть бути зібрані ключі шифрування, а в деяких випадках до того, як комп'ютер вимкнеться, може бути відображений логічний об'єм жорсткого диска (відомий як активне придбання).

Видалені файли - є загальною технікою, що використовується в комп'ютерній криміналістиці, є відновлення видалених файлів. Сучасне судово-медичне програмне забезпечення має власні інструменти для відновлення чи вилучення видалених даних. Більшість операційних систем та файлових систем не завжди стирають дані фізичних файлів, що дозволяє дослідникам реконструювати їх з секторів фізичного диска. Різання файлів передбачає пошук відомих заголовків файлів у зображенні диска та відновлення видалених матеріалів.

Стохастична криміналістика - метод, який використовує стохастичні властивості комп'ютерної системи для дослідження діяльності, де відсутні цифрові артефакти. Головним його використанням є вивчення крадіжки даних.

Стеганографія - однин з методів, які використовуються для приховування даних, процес приховування даних усередині цифрового зображення. Прикладом може бути приховати порнографічні зображення дітей або іншу інформацію, яку конкретний злочинець не хоче відкрити. Фахівці комп'ютерної судової експертизи можуть боротися з цим, переглянувши хеш файлу та порівнявши його з вихідним зображенням (якщо є.) Хоча зображення виглядає точно так само, хеш змінюється, коли дані змінюються.

Основні етапи комп'ютерної криміналістики:

- збір;
- дослідження;
- аналіз;
- подання.

На першому етапі відбувається збір як інформацією самої по собі, так і носіїв комп'ютерної інформації [5]. Збір повинен супроводжуватися поміткою, зазначенням джерел та походження даних і об'єктів. В процесі збору повинні забезпечуватися збереження і цілісність (незмінність) інформації, а в деяких випадках також її конфіденційність.

Два основних типи даних збираються в комп'ютерній криміналістиці. Постійні дані або не волотильні дані - це дані що зберігається на локальному жорсткому диску (або іншому носії) і зберігається коли комп'ютер вимкнений. Нестабільні дані або волотильні - це будь-які дані, які зберігаються в пам'яті або існують у транзит, який буде втрачено, коли комп'ютер втратить живлення або вимкнеться. Нестабільні дані знаходиться в регістрах, кеш-пам'яті та оперативної пам'яті (ОЗП).

Аналіз «живого» та аналіз вимкненої системи в принципі відрізняються не тільки підходами, але й даними, які вдалося зібрати. В першому випадку (Live Forensic) це спостереження за атакованою системою в реальному часі. Є

можливість безпосередньо оцінити дискову та мережеву активність, отримати список запущених процесів, відкритих портів, переглянути залогінених користувачів, а головне - отримати від оперативної пам'яті ті дані, які при відключенні живлення зникають безслідно.

Наприклад, ключі трояна-шифровальника, расшифровану копію смонтованих криптоконтейнерів і навіть доступ до віддалених ресурсів, якщо на них в поточному сеансі була виконана авторизація. Після вимкнення комп'ютера вся ця інформація буде безвідмовно втрачено, а кріптографічні контейнери залишаться зашифрованими файлами на диску.

Індивідуальні підходи у відділі розслідування комп'ютерних інцидентів антивірусної компанії (яка разом з правоохоронними органами аналізує нове випадок масового зараження) або FinCERT, що розриває APT в будь-якій платіжній системі. Тут же все другие приоритеты: треба зібрати максимум інформації, яку потім можна буде використовувати в суді. Обнаружить C & C-сервери ботнета, знайти джерело загрози, детально відновити сценарій підриву та вийти на його організаторів (в ідеалі - з їх подальшим арестом). Для цього потрібно тщательно зібрати доказну базу, але проблема в тому, що аналіз системи майже завжди вносить в неї розбещення. Завдання спеціаліста з комп'ютерної криміналістики - звести їх до мінімуму та враховувати при подальшому аналізі.

Окремі труднощі виникають з твердотільними накопичувачами [2]. Їх контролери самостійно виконують низькорівневі операції для прискорення подальшого запису і вирівнювання ступеня зносу осередків флеш-пам'яті. Банальні операції Garbage Collection і TRIM можуть виконатися в будь-який момент і поховати надію на відновлення видалених файлів. Варіанти вирішення цієї проблеми як мінімум два:

- Випаяти чіпи пам'яті і замінити їх программатором (марно, якщо на SSD використовується вбудоване шифрування, а зараз воно за замовчуванням активовано на більшості моделей).
- Створити довгу чергу команд на читання (звичайно контролер не виконує внутрішні операції, поки обробляє зовнішні запити).

Це ще одна причина підключати аналізований накопичувач вже після запуску ОС для криміналістичного аналізу і відразу знімати з нього по секторний образ, не залишаючи без дії ні секунди.

Додаткові спотворення при роботі з флеш-накопичувачами виникають через те, що ті осередки пам'яті, на які ні разу не виконувалася запис, знаходяться в невизначеному стані. Спроба вважати їх повертати не нулі, а випадкові дані - шум мікросхеми NAND Flash. Серед цього шуму просто за законом великих чисел можуть зустрітися байтові послідовності, що збігаються з відомими заголовками файлів. Тому програми відновлення даних при роботі в посекторному режимі (глибокий аналіз) часто «знаходять» вилучені файли там, де їх ніколи не було.

На другому етапі проводиться експертне дослідження зібраної інформації (об'єктів "носіїв). Воно включає зчитування інформації з носіїв, декодування і вичленення з неї тієї, яка відноситься до справи. Деякі дослідження можуть бути автоматизовані в тій чи іншій мірі. Але працювати головою і руками на цьому етапі експерту все одно доводиться. При цьому також повинна забезпечуватися цілісність інформації з досліджуваних носіїв.

На третьому етапі обрана інформація аналізується для отримання відповідей на питання, поставлені перед експертом або спеціалістом. При аналізі повинні використовуватися тільки наукові методи, достовірність яких підтверджена.

Четвертий етап включає оформлення результатів дослідження та аналізу в встановленої законом і в зрозумілій неспеціалістам формі.

Існують багато програмних продуктів які призначені для комп’ютерної криміналістики. Залежно від потреб, задач і бюджету можна вибрати певну програму. В наступному підрозділі будуть описані найпопулярніші програми для комп’ютерної криміналістики та їх недоліки.

1.3 Існуючі засоби для відтворення послідовності подій та їх недоліки

1.3.1 Платформа Autopsy

Autopsy - це платформа цифрової криміналістики і графічний інтерфейс для The Sleuth Kit і інших цифрових криміналістичних інструментів. Вона використовується правоохоронними органами, військовими і корпоративними експертами для розслідування того, що сталося на комп'ютерній системі. Звичайні користувачі можуть використовувати її, наприклад, для відновлення фотографій з цифрової карти пам'яті камери [8].

The Sleuth Kit - це бібліотека на мові C і набір інструментів командного рядка, які дозволяють досліджувати образи дисків. Ключова функціональність TSK дозволяє аналізувати томи і дані файлової системи в комп'ютері підозрюваного. Frameworks плагінів дозволяє імпортувати додаткові модулі для аналізу вмісту файлів і будувати автоматизовані системи. Бібліотека може бути імпортувати велику кількість інструментів цифрової криміналістики, а інструменти командного рядка можуть використовуватися безпосередньо для пошуку доказів.

Autopsy була створена щоб бути інтуїтивно зрозумілою з одразу після встановлення. Програма Autopsy запускає фонові завдання паралельно, використовуючи безліч ядер і виводить результати відразу після їх виявлення. На повне вивчення диска можуть піти годинник, але вже через хвилини, якщо ваші ключові слова були знайдені в призначеній для користувача домашньої папки, ви про це дізнаєтеся.

Програма була створена що б бути самостійною платформою з модулями, які поставляються при встановлення і доступні зі сторонніх джерел які можна в подальшому до інсталювати. Деякі з цих модулів забезпечують:

- Timeline Analysis (аналіз активності за часом) - високий рівень інтерфейс-графічного представлення активності в досліджуваній системі.

- Hash Filtering (фільтрація по Хешам) - Позначає файли, про які відомо, що вони погані, і ігнорує хороші файли.
- Keyword Search (пошук за ключовими словами) - індексованих пошук за ключовими словами для пошуку файлів, які згадують релевантні терміни.
- Web Artifacts (веб артефакти) - Витягує історію, закладки та cookies з Firefox, Chrome і IE.
- Data Carving (вирізка даних) - Відновлення видалених файлів з не розподіленого простору з використанням PhotoRec.
- Multimedia (мультимедіа) - Витягує EXIF з картинок і перегляд відео.
- Indicators of Compromise (індикатори компрометації) - Сканує комп'ютер з використанням STIX.

Інструмент Timeline Analysis організовується навколо певних подій. Подія має мітку часу, тип і опис. Усі події є дискретними, але вони можуть бути згруповані разом, щоб утворювати кластери з тривалістю у перегляді деталей в залежності від рівня опису, який увімкнено в інтерфейсі користувача.

Timeline Analysis має два режими відображення [9]. Перша - це стовпчаста діаграма, яка відповідає на питання про те, скільки даних відбулося за певний період часу. Використовуйте цей тип графіка, щоб показати, наскільки активність відбулася за певний період часу. Хоча це не покаже конкретних подій. Можливо, буде корисно визначити, коли останній користувався комп'ютером, або як часто він використовувався. Коли ви відкриваєте шкалу часу, вона відкриється у цьому стилі графіка (Рисунок 1.1).

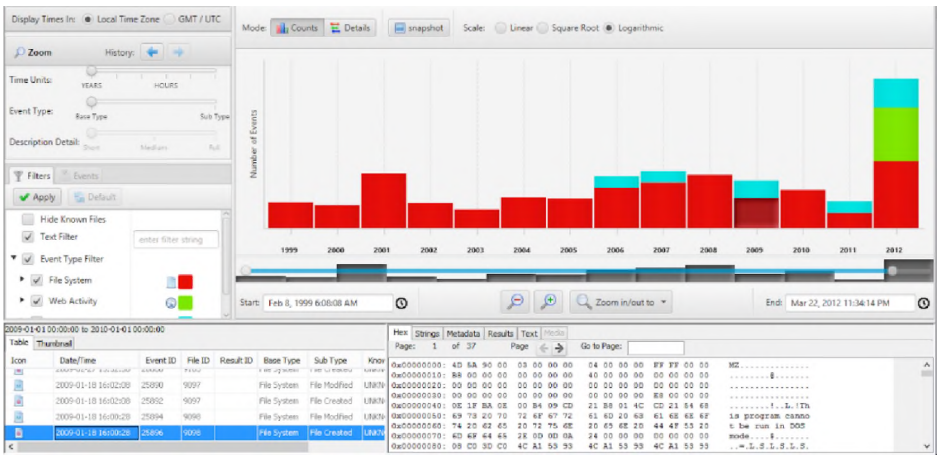

Рисунок 1.1 - Timeline Analysis інтерфейс відображення даних

Другий інтерфейс надає вам інформацію про події. Він має унікальний підхід до групування подібних подій разом, щоб запобігти перевантаженню даних. Багато часових поясів заважатимуть користувачеві при введенні даних з багатьох джерел, оскільки це занадто багато для розуміння. Autopsy має унікальний підхід кластеризації подій, так що, наприклад, всі файли в тій же папці відображаються як одна подія, і всі URL-адреси з одного домену відображаються як одна подія. Якщо користувач хоче переглянути додаткову інформацію про цю папку або домен, він може збільшити його. В іншому випадку він прихований (Рисунок 1.2).

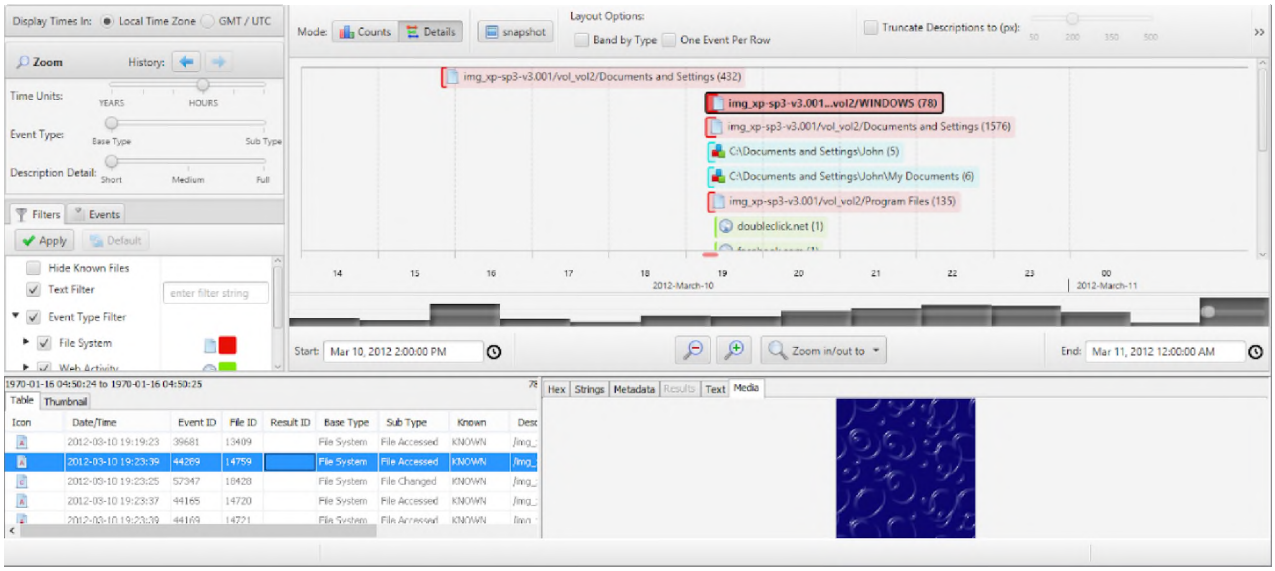

Рисунок 1.2 - Timeline Analysis другий варіант інтерфейсу відображення даних

Незалежно від режиму відображення, ви можете переглядати вміст файлу у різних глядачах та мати повний доступ до можливостей позначки з Autopsy.

Спільною проблемою з аналізом часової шкали є перевантаження інформації. Для цього в інтерфейсі Autopsy ви можете скористатись трьома способами масштабування, що допоможе вам визначити правильні дані. Їх можна керувати з однієї області у верхньому лівому куті інтерфейсу.

- Одиниці часу : цей рівень масштабування контролює тимчасові деталі, відображені на осі X. Це диктує, чи будуть маркери в масштабі років або секунд. Оскільки ви бажаєте отримати докладнішу інформацію про те, що

сталося в певний часовий проміжок, ви збільшите масштаб цього елемента керування.

- Тип події : цей рівень масштабування контролює, який тип типу події ви бачите. Як приклад, існує тип верхнього рівня "Файлова система" з підтипами для модифікованого часу, часу доступу та створених часів. Якщо ви хочете отримувати більш детальну інформацію про певний тип, то ви збільшите його за допомогою цього елемента керування.
- Подробиці опису : Цей рівень масштабування є найбільш унікальним для Autopsy та поділяє подібні події разом залежно від їх опису. Як приклад, він об'єднає об'єкти файлової системи разом, якщо вони знаходяться в тій самій кореневій папці, коли ви збільшуєте весь вихід. Це дозволяє загалом побачити, де відбувається активність, не бачити кожного окремого файлу.

Після тестування програмного продукту Autopsy я залишився задоволений. Основні переваги його це зручнй інтерфейс, велика палітра налаштувань залежно від потреб користувача, можливість експорту результатів у файл. В подальшому у своїй роботі я використовував деякі можливості Autopsy, зокрема зчитування не волотильних даних з системи та в подальшому їх використовував. Один з інструментів Autopsy Timeline Analysis який проводить аналіз активності даних за часом і будує послідовність подій, для потреб моєї роботи не зовсім підійшов. Він пропонує багато налаштувань та варіантів виводу даних користувачу але, основний недолік це те що Timeline Analysis обробляє лише не волотильні дані, що без волотильних даних не дає побудувати більш точну послідовність подій та побачити більш чітку картину подій. Також не зважаючи на зручність роботи в Timeline Analysis, побудова послідовності подій займає значну частину часу. Це звичайно можна усунити, використовувавши для тестування більш потужне обладнання, або придбати професійне обладнання для комп’ютерної криміналістики, але через відсутність матеріально технічної бази я не зміг це зробити.

1.3.2 Інструмент візуального аналізу IBM i2 Analyst's Notebook

IBM i2 Analyst's Notebook - це інструмент візуального аналізу, що дозволяє перетворити дані в цінну інформацію. Це рішення надає такі інноваційні можливості, як візуалізація підключених мереж, аналіз соціальних мереж, а також геопросторові або часові уявлення, які допомагають виявити приховані зв'язки і закономірності в даних.

Інтеграційні рішення i2 Analyst's Notebook дозволяють об'єднувати дані, вже накопичені в організації і рознесені по різних додатків, і проводити аналіз без додаткового завантаження, конвертації і перемикання між завданнями. Наявні в Analyst's Notebook аналітичні інструменти дозволяють виконувати пошук елементів на схемі, взаємозв'язків між ними і інші завдання. Analyst's Notebook дозволяє:

- Швидкий пошук спільних елементів і взаємозв'язків, прихованих в даних.
- Простота інтерпретації складної інформації.
- Широкий спектр доступних аналітичних технологій розслідування.
- Графічне відображення результатів, що робить їх більш доступними і простими для розуміння.
- Створення динамічних і наочних діаграм, що включають в себе фотографії, відеозаписи та документи.
- Зміна уявлення типів діаграм з діаграми зв'язків на діаграму часовій послідовності подій і навпаки.
- Легкість керування діаграмами і пошуку на них об'єктів, що цікавлять.
- Поширення діаграм в друкованому та електронному вигляді між зацікавленими сторонами.

Одним із прикладів використання продуктів сімейства IBM i2 може служити розслідування дій в інформаційній системі, який девайс та коли був підключений до системи. Для реалізації описаної завдання, за умови, що журнали системних подій збираються тільки з однієї системи, досить використання i2 Analyst's Notebook. Дані журналів аудиту завантажуються з текстових файлів в i2

Analyst's Notebook. Серед інформації, представленої в журналі, можна виділити різні типи об'єктів: сервери, телефони, периферійні пристрої. Отримані дані можна представити за допомогою різних видів діаграм. Наприклад, можна побудувати часову діаграму, що відображає на шкалі часу виникнення різних видів подій. Таким чином(Рисунок 1.3):

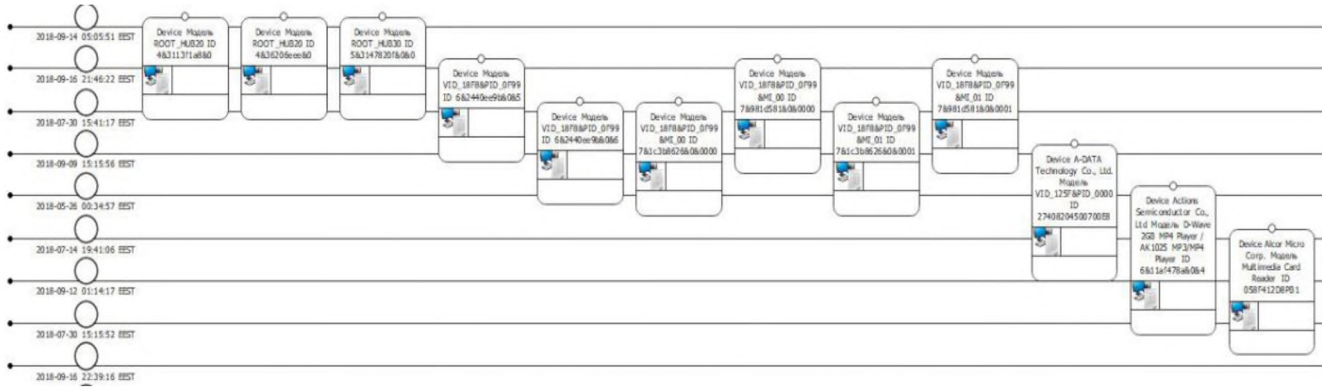

Рисунок 1.3 - IBM i2 приклад часової діаграми

Також наявну інформацію можна відобразити в діаграмі взаємозв'язків, яка дозволить відобразити кількість подій для кожної можливої зв'язку. Ця діаграма може допомогти в актуалізації прав доступу до систем і пошуку підозрілих активностей. Наприклад побудова діаграми взаємопов’язаних подій для інформації про веб пошук (Рисунок 1.4):

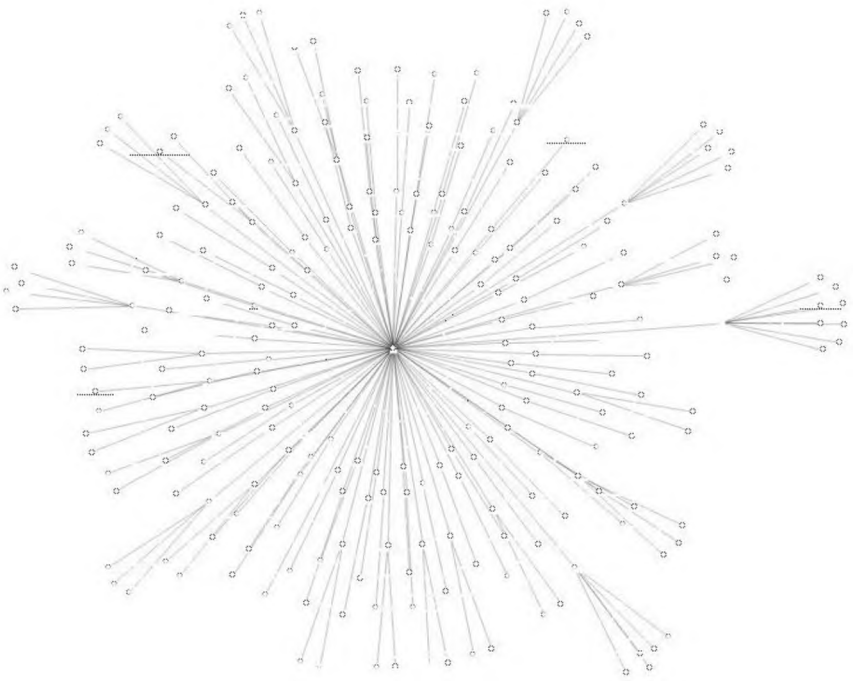

Рисунок 1.4 - IBM i2 приклад діаграми взаємопов’язаних подій

Натиснувши на кожну з вершин, користувач може отримати детальну інформацію про той чи інший об’єкт. Аналітичні функції i2 Analyst'sNotebook дозволяють знаходити на схемі «шляху», що з'єднують різні об'єкти, використовувати візуальний пошук серед об'єктів знаходяться на схемі, шукати об'єкти схожі з яких-небудь параметрами, редагувати і додавати нові об'єкти і зв'язку. Після завершення роботи аналітика схему можна зберегти у форматі i2, або вивести схему на друк. Основні переваги IBM I2 Analyst's Notebook:

- Наявність SDK для розширення функціональності
- Наочні схеми, багаті засоби візуального аналізу
- Потужні засоби аналізу
- Кластерний аналіз соціальних мереж
- Застосування статистики соціальних мереж і теплових карт, і ін.
- Великі обсяги різнорідних даних
- Вартість рішення.
- Простота настройки і використання.
- Можливість завантажувати дані з різних які пов'язані джерел.

Основні недоліки IBM I2 Analyst's Notebook:

- Неможливо підключитися безпосередньо до бази даних.
- Великі вимоги до продуктивності робочої станції.
- Низька працездатність при роботі з файлами великих обсягів.
- Неможливість обробки великих обсягів даних.
- Не можливість праці з великою кількістю різнотипних даних.

1.3.3 Інструмент ProDiscover Forensic

Комерційний інструмент для форензіки, який використовує власний формат образ файлу ProDiscover. ProDiscover може перетворити вихідний образ диску на завантажувальну машину VMWare. ProDiscover Forensic - це потужний інструмент комп'ютерного захисту, який дозволяє професіоналам знаходити всі

дані на диску комп'ютера і одночасно захищати докази та створювати якісні докази для використання в судових процесах. Використовуючи найкращі практичні поради та найменш руйнівний підхід до методології, ProDiscover Forensic дозволяє перевіряти файли без зміни цінних метаданих, таких як доступ до останнього часу. ProDiscover Forensic може:

- Відновити видалені дані;
- Отримати доступ до альтернативних потоків даних Windows;
- Дозволяє динамічно переглядати дані;
- Пошук та зчитування даних захищеної зони апаратного забезпечення на диску;

ProDiscover надає функціональні можливості, подібні до інших повнофункціональних судово-криміналістичних програмних комплексів, перелічених у цьому розділі. Технологічні шляхи також пропонує безкоштовну версію ProDiscover Basic. ProDiscover Basic - це комплект судово-криміналістичного програмного комплексу на базі повного GUI. Вона включає в себе можливість зображувати, зберігати, аналізувати та звітувати про докази, знайдені на комп'ютерному диску. Ця версія є безкоштовною, яка може бути використана та надається для роботи безкоштовно.

Основні переваги:

- Є безкоштовна версія продукту ProDiscover Basic
- Графічний інтерфейс
- Аналіз диску та створення його образу
- Гнучкий та швидкий інструмент для аналізу

Недоліки програмного продукту:

- Аналізує лише не волатильні дані
- В безкоштовній версії програми обмежений функціонал
- Обмежені налаштування в перегляді послідовності подій
- Відображає лише явні взаємозв’язки.

1.3.4 Програмне забезпечення Forensic Toolkit

Forensic Toolkit - програмне забезпечення для комп’ютерної криміналістики, розроблене AccessData [10]. Він сканує жорсткий диск, шукаючи різну інформацію. FTK - це платне програмне забезпечення, яке розроблене для швидкої аналітики та масштабованості в корпоративному класі. Відомий своїм інтуїтивно зрозумілим інтерфейсом, аналізом електронної пошти, має налаштування для перегляду даних та стабільністю роботи, FTK створює основу для безперервного розширення, тому розслідування може розширюватися з урахуванням потреб вашої організації.

Крім того, в AccessData пропонуються нові модулі розширення, що забезпечують найвищу продуктивність в аналізі шкідливих програм, а також найсучаснішу візуалізацію. Ці модулі інтегруються з FTK для створення найбільш повної платформи комп'ютерної криміналістики на ринку.

Одна з переваг продукту, це наявність візуалізації. Перегляд даних у кількох форматах відображення, включаючи часові шкали, графіки кластерів, кругові діаграми тощо (Рисунок 1.5). Швидко визначте відносини в даних, знайдіть основні елементи інформації та створюйте звіти.

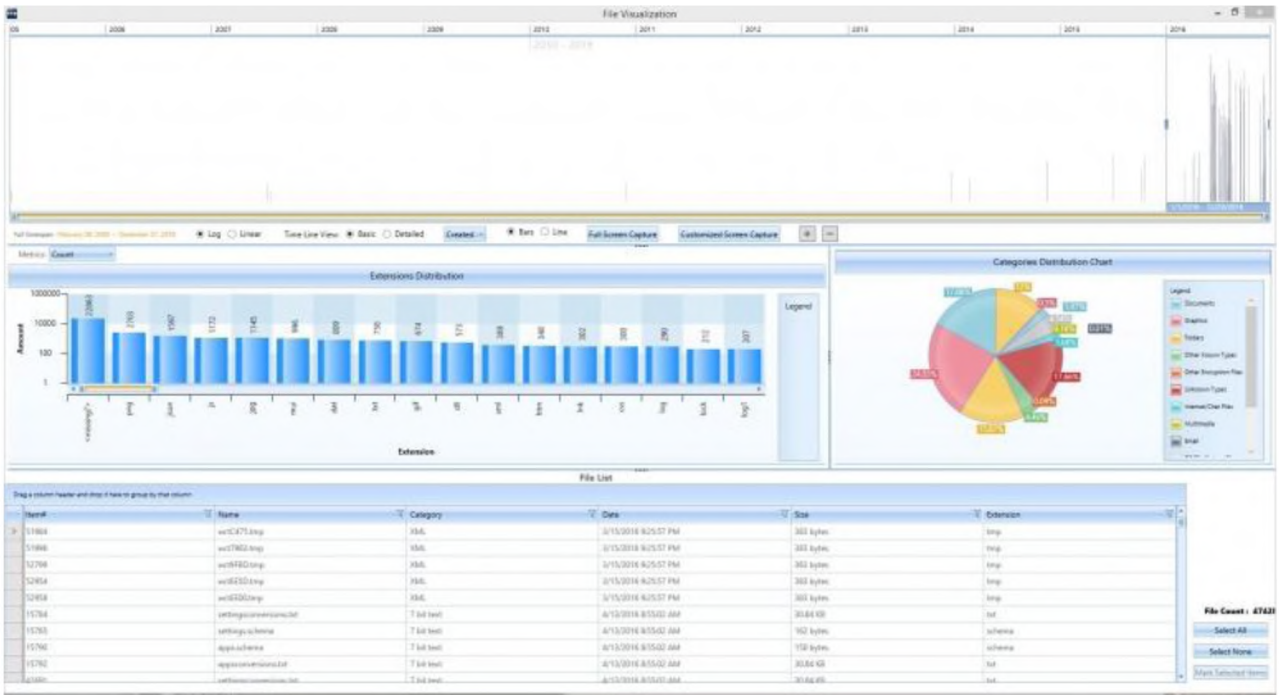

Рисунок 1.5 — інтерфейс Forensic Toolkit

Інструментарій також включає в себе автономну програму візуалізації диска під назвою FTK Imager. FTK Imager - це простий, але стислий інструмент. Це зберігає зображення жорсткого диска в одному файлі або в сегментах, які пізніше можуть бути реконструйовані. Він обчислює хеш-значення MD5 і підтверджує цілісність даних перед закриттям файлів. Результатом є файли зображень, які можна зберегти у кількох форматах, включаючи DD raw.

FTK відрізняється від інших програмних засобів тим що він обробляє дані авансом, так що ви не витрачати час на очікування пошуки на етапі аналізу. Тим не менш, продукт розроблений таким чином, щоб забезпечити найшвидшу, точнішу і послідовну судово-медичну обробку з можливістю розподіленої обробки та справжньої багатопотокової / багатоядерної підтримки. Кожна копія FTK включає в себе 4 оброблювачі - 1 на екзаменаційній машині та 3 - розподілені. Якщо ви зацікавлені в тому, щоб кілька екзаменаторів мали спільну обробку та централізовану базу даних для спільного аналізу, зе можна реалізувати за допомогою продуктів AccessData. Можливості FTK:

- Робота з майстром забезпечує відсутність даних.
- Скасувати / призупинити / відновити функціональність
- Статус обробки в реальному часі
- Дроселювання ресурсів ЦП
- Повідомлення електронною поштою після завершення обробки
- Доопрацювання до і після обробки
- Розширений функціонал дозволяє вказати критерії, такі як розмір файлу, тип даних та розмір пікселів, щоб зменшити кількість нерелевантних даних, вирізаних під час збільшення загального аналізу.

Загалом FTK є повноцінним професійним продуктом для форензіки з великою кількістю функціоналу та налаштуваннями. FTK використовує розподілену обробку і є єдиним рішенням для криміналістичної експертизи для повного використання багатоядерних комп'ютерів. Хоча інші інструменти судової експертизи витрачають потенціал сучасних апаратних рішень, FTK використовує

100 відсотків своїх апаратних ресурсів, допомагаючи слідчим швидше знаходити відповідні докази.

Оскільки індексування здійснюється вперед, фільтрація та пошук виконуються більш ефективно, ніж з будь-яким іншим рішенням. Незалежно від того, чи вивчаєте ви чи здійснюєте огляд документів, у вас є файл спільного індексу, що виключає необхідність відтворити чи дублювати файли.

FTK - справді керована базами даних, використовуючи одну загальну базу даних для розслідування. Усі дані зберігаються надійно та централізовано, що дозволяє вашим командам використовувати однакові дані. Це зменшує вартість та складність створення кількох наборів даних.

Основним недоліком на мій погляд є те що FTK аналізує лише дані з жорсткого носія и не зчитує тимчасові дані, не виявляє можливі взаємозв'язки а лише явні, FTK має тільки платне програмне забезпечення і для аналізу коли бюджет не великий це є основним критерієм на мою думку.

1.3.5 Digital Forensics Framework

Digital Forensics Framework - це криміналістична комп'ютерна платформа із відкритим вихідним кодом, він побудований на окремому API [11]. DFF прийшов на заміну застарілим цифровим криміналістичним програмним продуктам, що використовується сьогодні. Створений для простого використання та автоматизації, інтерфейс DFF демонструє користувачу основні кроки під час цифрового розслідування, тому він може використовуватися як професіоналами, так і не професіоналами для швидкого та простого здійснення цифрових розслідувань та реагування на інциденти.

DFF здатен виконати швидкий аналіз диска та швидко отримує доступ до пам'яті, глибоке дослідження комп'ютерів або смартфонів. DFF використовує технології блокового запису, щоб захистити доказ і зберегти цілісність медіа. DFF здатен агрегувати різні інформаційні джерела, з волатильної пам'яті. Це дозволяє

вам мати уявлення про події в системі та користувацьку активність. Основні можливості DFF: збереження доказів, запис логічних блоків, аналіз сирих форматів, сумісність з файловим форматом Encase EWF, сумісність з файловим форматом AFF, відстеження (ланцюжків зберігання), розрахунок криптографічних хеш-кодувань, швидке скорочення і сортування даних, виявлення сигнатур файлів, високий рівень фільтрації і пошуку, реконструкція томів і файлової системи, виявлення і монтування розділів, формат віртуального диска VMDK - FAT 12/16/32 (Thumbdrive), NTFS з ADS і підтримкою стиснення (Microsoft Windows), HFS HFS + HFSx файлові системи (OS X & iphone), Ext2 / 3/4 файлові системи (GNU / Linux і Android), аналіз мультимедіа, галерейний перегляд, виділення метаданих EXIF, аналіз Windows OS, парсер файлів LNK, аналіз Prefetchcta регістра, поштові скриньки Microsoft Outlook PST, аналіз пам'яті, Інтеграція фреймворка Volatility, графічна реконструкція дерева процесів (суміш pstree і psxview), інформація про процеси (підключення, procdump), VAD доступ з RWX сторінками поміченими як підозрілі, аналіз документів та виділені переглядачі (PDF, Тексти, Веб)

DFF є безкоштовним програмним забезпеченням який можна використовувати безпосередньо з інтерпретатора Python, що на мою думку є дуже зручним для розробників додатків які планують в подальшому використовувати зібрані дані. Нажаль DFF має помітно менше налаштувань ніж в інших програмах, та має більш примітивний варіант відображення подій.

Висновки до розділу 1

В цьому розділі було розглянуто сучасні проблеми комп’ютерної криміналістики, та описано принцип відновлення подій в інформаційній системі. Також розглянуто існуючі програмні продукти для відновлення подій в інформаційній системі. У розглянутих програмних продуктів є певні недоліки в їх роботі, наприклад:

- Аналіз лише певної групи даних, волатильних або не волатильних;
- Швидкість роботи;
- В деяких програмних продуктів обмежений функціонал;
- Пошук лише явних взаємозв’язків;
- Обмежена кількість даних які можна обробляти;
- Представлення послідовності подій лише у вигляді TimeLine;
- Відсутній пошук взаємопов’язаних даних за певним параметром.

Через ці недоліки користувач який використовує дані програмні продукти не зможе відновити повну послідовність подій а лише часткову. Таким чином після аналізу даних його висновки можуть бути хибними, або не дадуть бажаного результату зовсім. Тому я вирішив реалізувати власний метод відновлення ланцюгів подій в інформаційній системі.

2 ВИКОРИСТАННЯ МЕТОДІВ ТЕОРІЇ ГРАФІВ ДЛЯ ВІДНОВЛЕННЯ ЛАНЦЮГІВ ПОДІЙ

В даному розділі будуть розглянуті сучасні методи теорії графів для відновлення ланцюгів подій. Буде описано їх ідею та принцип роботи а також швидкість їх роботи. Буде визначено метод пошуку шляхів в графі між вершинами який в подальшому буде використовуватися в програмі.

Для пошуку взаємопов'язаних подій я використовую у своїй роботі теорію графів. Всі події я представляю як вершини графу, кожна подія має список своїх параметрів. Ребрами в моєму графі є спільні параметри двох вершин. Тому для пошуку взаємопов'язаних подій треба вибрати метод пошук шляхів в графі між вершинами. Далі представлені основні методи для пошуку шляхів в графі.

2.1 Пошук в глибину

Пошук в глибину - один з методів обходу графа. Стратегія пошуку в глибину, як і випливає з назви, полягає в тому, щоб йти «вглиб» графа, наскільки це можливо. Алгоритм пошуку описується рекурсивно: перебираємо всі вихідні з даної вершини ребра. Якщо ребро веде в вершину, яка не була розглянута раніше, то запускаємо алгоритм від цієї нерозглянутих вершини, а після повертаємося і продовжуємо перебирати ребра. Повернення відбувається в тому випадку, якщо в даній вершині не залишилося ребер, які ведуть в нерозглянутих вершину. Якщо після завершення алгоритму не всі вершини були розглянуті, то необхідно запустити алгоритм від однієї з нерозглянутих вершин.

Загальна ідея алгоритму полягає в наступному: для кожної не проторений вершини необхідно знайти всі не пройдені суміжні вершини і повторити пошук для них. Нехай дано граф, наведений на (Рисунок 2.1).

Суть алгоритму полягає в наступному: на кожному кроці для поточної вершини перебираємо всі суміжні з нею вершини і для кожної з них рекурсивно повторюємо всі дії, при цьому запам'ятовуємо в деякій структурі, які вершини вже були відвідані. Тобто з поточної вершини йдемо в першу суміжну з нею, але поки що відвідані, до тих пір, поки це можливо. Якщо для поточної вершини немає суміжних або вони були всі переглянуті, то повертаємося на рівень вгору (до попередньої вершині) і продовжуємо перебір суміжних з нею вершин.

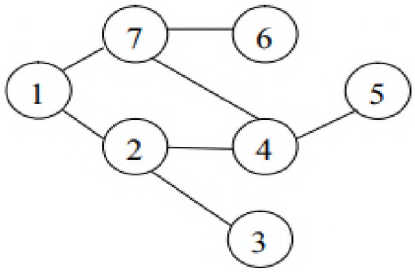

Рисунок 2.1 - Приклад графу

Припустимо, що ми почали пошук з першої вершини, порядок відвідування вершин для графа на рис. 2 буде таким: 1 2 3 4 5 7 6. Якщо запустити обхід графа з вершини 4, то порядок обходу буде наступним: 4 2 1 7 6 3 5.

Оскільки ми використовували матрицю суміжності для завдання графа, то оцінка часової складності даного алгоритму буде O (N^2): нам потрібно відвідати N вершин, і для кожної вершини ми перебираємо N вершин. Можна реалізувати і нерекурсивний варіант даної процедури, для цього доведеться використовувати структуру даних стек для запам'ятовування порядку обходу вершин.

Для пошуку всіх шляхів між двома вершинами скористатися пошуком в ширину вже не вийде, тут можна використовувати тільки пошук в глибину, який дозволяє після досягнення кінцевої вершини, повернутися на крок назад і спробувати знайти інший шлях.

Припустимо, ми хочемо знайти всі можливі шляхи з вершини 3 в вершину 6. Якщо не розглядати шляхи, що містять цикли, то все існує чотири шляхи:32176, 3217456, 32456,32476.

2.2 Метод перебору

Метод перебору - невелика модифікація алгоритму обходу в глибину. Запустимо обхід в глибину від вершини s. При кожному відвідуванні вершини v перевіримо, чи не є вона шуканої вершиною t. Якщо це так, то відповідь збільшується на одиницю і обхід припиняється. В іншому випадку проводиться запуск обходу в глибину для всіх вершин, в які є ребро з v, Причому він проводиться незалежно від того, були ці вершини відвідані раніше, чи ні.

Час роботи даного алгоритму в гіршому випадку O(Ans), Де Ans- число шляхів в графі з s в t. Наприклад, на наступному графі даний алгоритм буде мати час роботи $O(2^{(n/2)})$ (Рисунок 2.2). Якщо ж використовувати метод динамічного програмування, мова про який піде нижче, то асимптотику можна поліпшити до O(n).

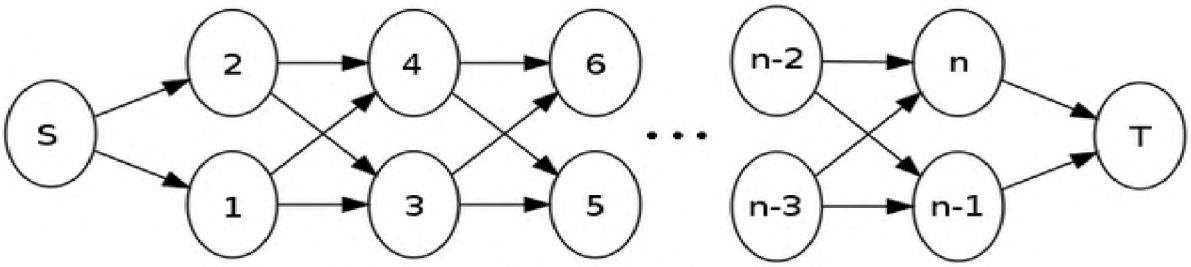

Рисунок 2.2 — Приклад графу

Приклад роботи методу:

Розглянемо приклад роботи алгоритму на наступному графі (Рисунок 2.3):

Спочатку масиви d і w ініціалізовані наступним чином (Таблиця 2.1):

Таблиця 2.1 - Початковий масив

Вершина	S	1	2	3	4	T
w	true	false	false	false	false	false
d	1	0	0	0	0	0

Спочатку функція count буде викликана від вершини T. Відповідь для неї ще не пораховано (w[T]=false), Отже count буде викликана від вершин 3 і 4. Для вершини 3 відповідь теж не пораховано ([3]=false), Отже count буде викликана вже для вершин 2 і S. А ось для них відповідь ми вже можемо дізнатися: для 2 він дорівнює d[S], Так як це S - єдина вершина, ребро з якої входить в неї. Безпосередньо для S відповідь нам також відомий. На поточний момент таблиця буде виглядати наступним чином (Таблиця 2.2):

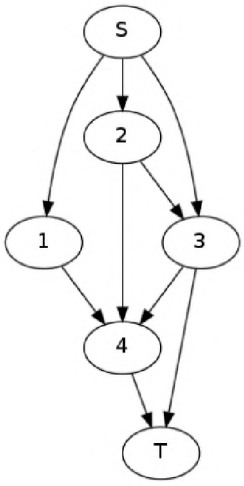

Рисунок 2.3 - Приклад графу

Таблиця 2.2 - Масив значень

вершина	S	1	2	3	4	T
w	true	false	true	false	false	false
d	1	0	1	0	0	0

Тепер ми знаємо значення для вершин 2 і S, Що дозволяє обчислити d[3]=d[2]+d[S]=2. Також оновимо значення в масиві w (Таблиця 2.3): w[3] = true.

Таблиця 2.3 – Масив значень

вершина	S	1	2	3	4	T
w	true	false	true	true	false	false
d	1	0	1	2	0	0

На самому початку для обчислення d[T] нам були потрібні значення d[3] і d[4]. Тепер нам відомо значення d[3], тому простежимо за тим, як буде обчислюватися d[4]. d[4]=count(3)+count(2)+count(1), але w[3]=true, w[2]=true, Отже значення d[3] і d[2] ми вже знаємо, і нам необхідно викликати count(1). Відповідь для цієї вершини дорівнює d[S], Так як це єдина вершина, ребро з якої входить в 1. Оновимо відповідні значення масивів d і w (Таблиця 2.4):

Таблиця 2.4 – Масив значень

вершина	S	1	2	3	4	T
w	true	true	true	true	false	false
d	1	1	1	2	0	0

Тепер нам відомі всі три значення, що вимагаються для обчислення відповіді для ершини 4. d[4]=d[3]+d[2]+d[1]=2+1+1=4 (Таблиця 2.5):

Таблиця 2.5 – Масив значень

вершина	S	1	2	3	4	T
w	true	true	true	true	true	false
d	1	1	1	2	4	0

Далі, обчислимо d[T] = d[3] + d[4] = 2 + 4 = 6 і оновимо таблиці d і w (Таблиця 2.6):

Таблиця 2.6 – Фінальний масив значень

вершина	S	1	2	3	4	T
w	true	true	true	true	true	true
d	1	1	1	2	4	6

Цей алгоритм дозволяє обчислити кількість шляхів від будь-якої вершини S не тільки до T, але і для будь-якої вершини, що лежить на будь-якому з шляхів від S до T. Для цього достатньо взяти значення у відповідній клітинці d.

2.3 Метод пошуку в ширину

Метод пошуку в ширину - це один з основних алгоритмів на графах. В результаті пошуку в ширину знаходиться шлях найкоротшої довжини в невиважені графі, тобто шлях, що містить найменше число ребер. Алгоритм працює за $O(n+m)$, де n- число вершин, m- число ребер [4].

Опис алгоритму: на вхід алгоритму подається заданий граф (незважений), і номер стартовою вершини s. Граф може бути як орієнтованим, так і неорієнтованим, для алгоритму це не важливо.

Сам алгоритм можна розуміти як процес "підпалювання" графа: на нульовому кроці підпалюємо тільки вершину s. На кожному наступному кроці вогонь з кожної вже палаючої вершини перекидається на всіх її сусідів; тобто за одну ітерацію алгоритму відбувається розширення "кільця вогню" в ширину на одиницю (звідси і назва алгоритму).

Більш строго це можна представити таким чином. Створимо чергу q, в яку будуть міститися палаючі вершини, а також заведемо булевский масив $\mathrm{used}[]$, в якому для кожної вершини будемо відзначати, горить вона вже чи ні (або іншими словами, чи була вона переглянуло).

Спочатку в чергу поміщається тільки вершина s, і $\mathrm{used[s]} = \mathrm{true}$, а для всіх інших вершин $\mathrm{used}[] = \mathrm{false}$. Потім алгоритм являє собою цикл: поки черга не порожня, дістати з її голови одну вершину, переглянути всі ребра, що виходять з цієї вершини, і якщо якісь з переглянутих вершин ще не горять, то підпалити їх і помістити в кінець черги.

У підсумку, коли черга спорожніє, обхід в ширину обійде всі досяжні з s вершини, причому до кожної дійде найкоротшим шляхом. Також можна

порахувати довжини найкоротших шляхів (для чого просто треба завести масив довжин шляхів $d[]$), і компактно зберегти інформацію, достатню для відновлення всіх цих найкоротших шляхів (для цього треба завести масив "предків" $p[]$, в якому для кожної вершини зберігати номер вершини, по якій ми потрапили в цю вершину).

Варіації алгоритму:

- BFS 0-1: Нехай в графі дозволені ребра ваги 0 і 1, Необхідно знайти найкоротший шлях між двома вершинами. Для вирішення даного завдання модифікуємо наведений вище алгоритм наступним чином:

  Замість черзі будемо використовувати грудня (або можна навіть steque). Якщо розглядається її ребро має вагу 0, То будемо додавати вершину в початок, а інакше в кінець. Після цього додавання, додатковий введений інваріант в доказі розташування елементів в деці в порядку неспадання залишиться активним, тому порядок в деці зберігається. І, відповідно, релаксуючи відстань до всіх суміжних вершин і, при успішній релаксації, додаємо їх в грудні. Таким чином, на початку дека завжди буде вершина, відстань до якої менше або дорівнює відстані до інших вершин дека, і інваріант розташування елементів в деці в порядку неспадання зберігається. Значить, алгоритм коректний на тій же підставі, що і звичайний BFS. Очевидно, що кожна вершина увійде в грудні не більше двох разів, значить, асимптотика у даного алгоритму та ж, що і у звичайного BFS.
- BFS 1-k: Нехай в графі дозволені ребра цілочислові ваги з відрізка 1...k1...k, Необхідно знайти найкоротший шлях між двома вершинами. Уявімо ребро uv ваги m як послідовність ребер $uu_1u_2 \dots u_m - 1^v$ (де $u_1 \dots u_{m-1}$ - нові вершини). Застосуємо дану операцію до всіх ребрах графа G. Отримаємо граф, що складається (в гіршому випадку) з k|E| ребер і |V| +(K-1) | E| вершин. Для знаходження найкоротшого шляху слід запустити BFS на новому графі. Даний алгоритм буде мати асимптотику O(|V|+k|E|).

2.4 Алгоритм дейкстри

Алгоритм дейкстри - алгоритм на графах, винайдений нідерландським вченим Едсгер Дейкстрой в 1959 році. Знаходить найкоротші шляхи від однієї з вершин графа до всіх інших [3]. Алгоритм працює тільки для графів без ребер негативного ваги. Складність алгоритму Дейкстри складається з двох основних операцій: час знаходження вершини з найменшою величиною відстані $d[v]$, і час здійснення релаксації, тобто час зміни величини $d[to]$.

При простій реалізації ці операції зажадають відповідно $O(n)$і $O(1)$часу. З огляду на, що перша операція виконується $O(n)$раз, а друга - $O(m)$, отримуємо асимптотики найпростішої реалізації алгоритму Дейкстри: $O(n^2 + m)$.

Зрозуміло, що ця асимптотика є оптимальною для щільних графів, тобто коли $m \approx n^2$ чим більше розріджене граф (тобто чим менше mв порівнянні з максимальним кількість ребер n^2), тим менше оптимальної стає ця оцінка, і з вини першого доданка. Таким чином, треба покращувати час виконання операцій першого типу, не сильно погіршуючи при цьому час виконання операцій другого типу.

Для цього треба використовувати різні допоміжні структури даних. Найбільш привабливими є фібоначчійовий купи, які дозволяють виробляти операцію першого виду за $O(\log n)$, а другого - за $O(1)$. Тому при використанні фібоначчійовий куп час роботи алгоритму Дейкстри складе $O(n \log n + m)$, Що є практично теоретичним мінімумом для алгоритму пошуку найкоротшого шляху. До речі кажучи, ця оцінка є оптимальною для алгоритмів, заснованих на алгоритмі Дейкстри, тобто Фібоначчійовий купи є оптимальними з цієї точки зору (це твердження про оптимальність насправді засноване на неможливості існування такої "ідеальної" структури даних - якби вона існувала, то можна було б виконувати сортування за лінійний час, що, як відомо, в загальному випадку неможливо; втім, цікаво, що існує алгоритм Торупа (Thorup), який шукає найкоротший шлях з оптимальною, лінійної, асимптотикою, але заснований він на

зовсім іншу ідею, ніж алгоритм Дейкстри, тому ніяких суперечностей тут немає). Однак, фібоначчійовий купи досить складні в реалізації (і, треба зазначити, мають чималу константу,

Як компроміс можна використовувати структури даних, що дозволяють виконувати обидва типи операцій (фактично, це витяг мінімуму і оновлення елемента) за $O(\log n)$. Тоді час роботи алгоритму Дейкстри складе: $O(m \log n + m \log m) = O(m \log n)$.

Приклад роботи методу:

Розглянемо виконання алгоритму на прикладі графа, показаного на малюнку (Рисунок 2.4).Нехай потрібно знайти найкоротший відстані від 1-ї вершини до всіх інших.

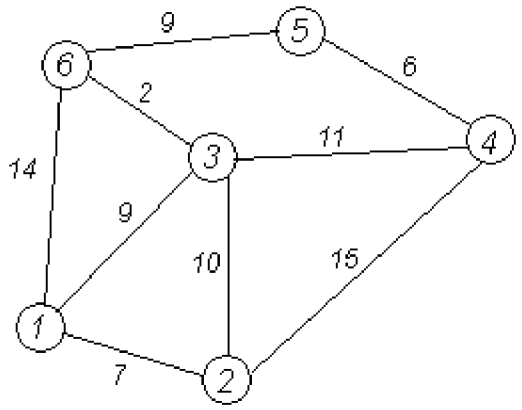

Рисунок 2.4 - Приклад графу

Колами позначені вершини, лініями - шляху між ними (ребра графа). У колах позначені номери вершин, над ребрами позначений їх вага - довжина шляху.

Перший крок: Розглянемо крок алгоритму Дейкстри для нашого прикладу. Мінімальну мітку має вершина 1. Її сусідами є вершини 2, 3 і 6.

Перший по черзі сусід вершини 1 - вершина 2, тому що довжина шляху до неї мінімальна. Довжина шляху в неї через вершину 1 дорівнює сумі значення мітки вершини 1 і довжини ребра, що йде з 1-й в 2-ю, тобто 0 + 7 = 7. Це менше поточної мітки вершини 2, нескінченності, тому нова мітка 2-й вершини дорівнює 7.

Аналогічну операцію проробляємо з двома іншими сусідами 1-й вершини - 3-й і 6-й. Всі сусіди вершини 1 перевірені. Поточне мінімальна відстань до вершини 1 вважається остаточним і перегляду не підлягає. Викреслимо її з графа, щоб відзначити, що ця вершина переглянуло.

Другий крок. Крок алгоритму повторюється. Знову знаходимо «найближчу» з невідвіданих вершин. Це вершина 2 з міткою 7 (Рисунок 2.5).

Знову намагаємося зменшити мітки сусідів обраної вершини, намагаючись пройти в них через 2-ю вершину. Сусідами вершини 2 є вершини 1, 3 і 4.

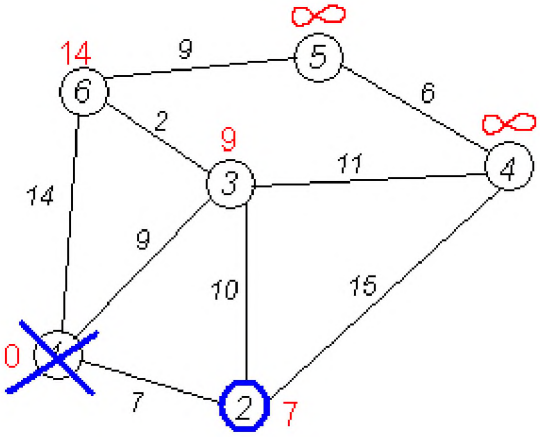

Рисунок 2.5 - Приклад графу на початку 2 кроку

Перший (по порядку) сусід вершини 2 - вершина 1. Але вона вже переглянуло, тому з 1-й вершиною нічого не робимо.

Наступний сусід вершини 2 - вершина 3, так як має мінімальну позначку з вершин, позначених що не відвідані. Якщо йти в неї через 2, то довжина такого шляху буде дорівнює 17 (7 + 10 = 17). Але поточна мітка третьої вершини дорівнює 9, а це менше 17, тому мітка не змінюється.

Ще один сусід вершини 2 - вершина 4. Якщо йти в неї через 2-ю, то довжина такого шляху буде дорівнює сумі найкоротшої відстані до 2-ї вершини і відстані між вершинами 2 і 4, тобто 22 (7 + 15 = 22). Встановлюємо мітку вершини 4 рівній 22.

Всі сусіди вершини 2 переглянуті, заморожуємо відстань до неї і помічаємо її як відвіданих.

Третій крок. Повторюємо крок алгоритму, вибравши вершину 3. Після її «обробки» отримаємо такі результати:

Подальші кроки. Повторюємо крок алгоритму для решти вершин. Це будуть вершини 6, 4 і 5, відповідно до порядку.

Завершення виконання алгоритму. Алгоритм закінчує роботу, коли не можна більше обробити жодної вершини. В даному прикладі всі вершини закреслено, проте помилково вважати, що так буде в будь-якому прикладі - деякі вершини можуть залишитися не зачеркнутими, якщо до них не можна дістатися, тобто якщо граф незв'язних. Результат роботи алгоритму видно на останньому малюнку: найкоротший шлях від вершини 1 до 2-ї становить 7, до 3-й - 9, до 4-ї - 20, до 5-ї - 20, до 6-ї - 11.

2.5 Алгоритм A*

Алгоритм A * - це алгоритм, який дозволяє знайти найкоротший шлях від одного вузла до іншого. Даний алгоритм є розширенням алгоритму Дейкстри, прискорення роботи досягається за рахунок евристики - при розгляді кожної окремої вершини перехід робиться в ту сусідню вершину, можливий шлях з якої до шуканої вершини найкоротший. Він оцінює поточний вузол, і якщо це не кінцевий вузол, він знаходить всі вузли, зв'язані з поточного вузла, оцінює їх відстань до кінцевого вузла, встановлює поточний вузол як головний вузол і додає їх до списку, який називається відкритим список [12].

Потім відкритий список опитаних для вузла, який оцінюється як найближчий до кінця, і повторює цей процес до відкриття кінцевого вузла або відсутність у вузлі інших версій вузлів, у цьому випадку неможливо знайти шлях.

Найкращим алгоритмом для пошуку оптимальних шляхів в різних просторах є A *. Цей евристичний пошук сортує всі вузли по наближенню найкращого маршруту йде через цей вузол. Типова формула евристики виражається у вигляді (2.1):

$$f(n) = g(n) + h(n) \quad (2.1)$$

де:

- f (n) значення оцінки, призначений вузлу n
- g (n) найменша вартість прибуття в вузол n з точки старту
- h (n) евристичне наближення вартості шляху до мети від вузла n

Поведінка алгоритму сильно залежить від того, яка евристика використовується. У свою чергу, вибір евристики залежить від постановки задачі. Часто A* використовується для моделювання переміщення по поверхні, покритої координатної сіткою (Рисунок 2.6).

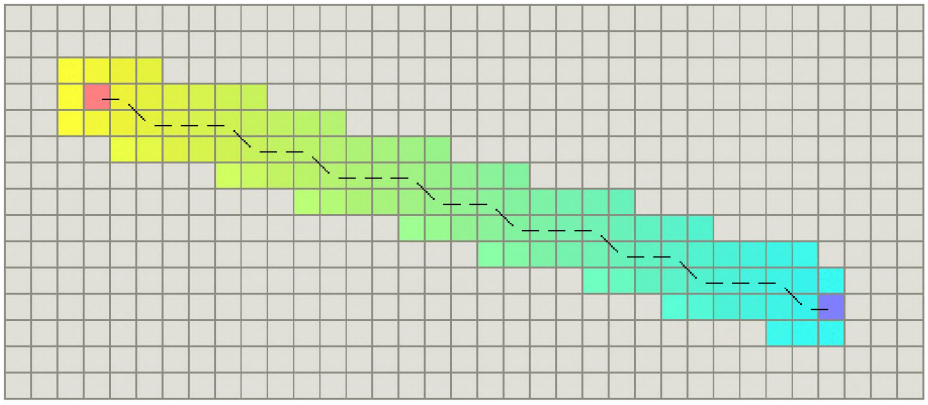

Рисунок 2.6 - Приклад A* на сітці з можливістю ходити в восьми напрямках

- Якщо ми можемо переміщатися в чотирьох напрямках, то в якості евристики варто вибрати Манхеттенську відстань (2.2):

$$h(v) = |v.x - goal.x| + |v.y - goal.y|. \quad (2.2)$$

Відстань Чебишева застосовується, коли до чотирьох напрямках додаються діагоналі (2.3):

$$h(v) = max(|v.x - goal.x|, |v.y - goal.y|). \quad (2.3)$$

- Якщо пересування не обмежена сіткою, то можна використовувати евклідова відстань по прямій (2.4):

$$h(v) = \sqrt{(v.x - goal.x)^2 + (v.y - goal.y)^2} \quad (2.4)$$

Також варто звернути увагу на те як співвідносяться f(v)і h(v). Якщо вони вимірюються в різних величинах (наприклад, g(v)- це відстань в кілометрах, а h(v) - оцінка часу шляху в годинах) A * може видати некоректний результат.

Тимчасова складність алгоритму A * залежить від евристики. У гіршому випадку, число вершин, досліджуваних алгоритмом, зростає експоненціально в порівнянні з довжиною оптимального шляху, але складність стає полиномиальной, коли евристика задовольняє наступній умові (2.5):

$$|h(x) - h^*(x)| \leq O(\log h^*(x)); \quad (2.5)$$

де h * - оптимальна евристика, тобто точна оцінка відстані з вершини x до мети. Іншими словами, помилка *h (x)* не повинна зростати швидше, ніж логарифм від оптимальної евристики.

Таким чином, цей алгоритм поєднує в собі облік довжини попереднього шляху з алгоритму Дійкстра з евристикою з алгоритму "кращий-перший". Алгоритм добре відображений в лістингу 3. Так як деякі вузли можуть оброблятися повторно (для пошуку оптимальних шляхів до них пізніше) необхідно ввести новий список Closed для їх відстежування. A * має безліч цікавих властивостей. Він гарантовано знаходить найкоротший шлях, до тих пір поки евристичне наближення h (n) є допустимим, тобто він ніколи не перевищує дійсного залишився відстані до цілі. Цей алгоритм найкращим чином використовує евристику: жоден інший алгоритм не розкриється менше число вузлів, не враховуючи вузлів з однаковою вартістю.

В результаті було розглянуто сучасні методи пошуку шляхів в графі. Було описано їх ідею та принцип роботи а також швидкість їх роботи та недоліки в їх роботі. Серед усіх методів пошуку зв'язку між вершинами найбільш оптимально підійшов алгоритм A *. Цей алгоритм продемонстрував швидку роботу та э дуже гнучким в використанні.

Висновки до розділу 2

В даному розділі було розглянуто сучасні методи теорії графів для відновлення ланцюгів подій. Було описано їх ідею та принцип роботи а також швидкість їх роботи. Було визначено метод пошуку шляхів в графі між вершинами який в подальшому буду використовувати в методі пошуку

взаємопов’язаних подій. Це алгоритм A*, він продемонстрував швидку роботу та э дуже гнучким в використанні. Сам метод, пошуку взаємопов’язаних подій в інформаційній системі, буде описаний в наступному розділі.

3 РЕАЛІЗАЦІЯ МЕТОДУ ПОШУКУ ВЗАЄМОПОВ'ЯЗАНИХ ПОДІЙ

В даному розділі буде реалізовано та докладно описано метод пошуку взаємопов'язаних подій, буде побудовано граф залежності та пошуку зв'язку між двома подіями за допомогою алгоритму A*. Також в даному розділі буде описано засоби які використовувалися для реалізації програми.

3.1 Опис реалізованого методу

Після ознайомлення з програмними продуктами для форензіки які зчитують дані з інформаційної системи та відновлюють послідовність подій, я виявив як і переваги так і недоліки в їх роботі. Ці програмні додатки давно представлені на ринок, та мають потужний функціонал, візуалізований інтерфейс та низку налаштувань які може вказати користувач залежно від своїх потреб. Вони є дуже зручними в використовувані але в них є і свої недоліки. Основні недоліки готових програмних додатків:

- Ціна програмного продукту та відсутність безкоштовної версії програми
- Аналіз не повної групи даних, аналіз або волотильних або не волотильних даних
- Побудова графу взаємозв'язків лише для одного типу даних
- Виявлення лише явних взаємозв'язків
- Побудова послідовності події лише за часовою шкалою
- Відсутня можливість пошуку взаємозв'язків

Аналізуючи недоліки представлених програм я вирішив реалізувати власну програму для відновлення подій в інформаційній системі, з урахуванням недоліків вже існуючих програмних додатків. Більшість програмних продуктів будують лише послідовність подій за часовою шкалою, але такий підхід не відображає картину взаємозв'язків повністю, адже певні події можуть викликати процеси або задачі через якийсь проміжок часу. В моєму програмному додатку реалізовано як

і послідовність подій за часовою шкалою, так і взаємозв’язки подій. Для виявлення взаємозв’язків я використовував теорію графів.

Теорія графів - у вузькому сенсі - розділ дискретної математики, одна з гілок дискретної топології, в широкому сенсі - теорія мереж, наука про топологічних формах, мережевих моделях уявлення будь-якого процесу або системи. Основним об'єктом дослідження цієї теорії є графи. Графом називають геометричну схему, що представляє собою систему ліній, що пов'язують якісь задані точки. Точки звані вершинами, а зв'язують їх лінії - ребрами (або дугами). Теорія графів обґрунтовує способи побудови графів, що виражають залежності або зв'язку в формі геометричних схем між різними одиницями тієї чи іншої сукупності. Фактично теорія графів є сукупність способів топологічних уявлень будь-яких процесів або структур.

Неорієнтованим графом називається безліч як завгодно розміщених на площині, точок, деякі з яких з'єднані лініями будь-якої форми. Два неорієнтованих графа вважаються невиразними, якщо вони відрізняються один від одного тільки формою сполучних ліній або способом розміщення точок на площині.

Неорієнтовані граф, G-граф - це впорядкована пара G: = (V, E), для якої виконані наступні умови:

- V - це непорожня безліч вершин, або вузлів,
- E - це безліч пар (в разі неорієнтованого графа - невпорядкованих) вершин, званих ребрами.
- V (а значить і, E, інакше воно було б мультімножество) зазвичай вважаються кінцевими множинами. Багато хороші результати, отримані для кінцевих графів, невірні (або будь-яким чином відрізняються) для нескінченних графів. Це відбувається тому, що ряд міркувань стає хибним в разі нескінченних множин.
- Вершини і ребра графа називаються також елементами графа.
- Порядок графа - це число вершин в графі | V |.
- Розмір графа - це число його ребер | E |

В якості вершин графа виступає конкретні події одних із типів даних які були зібрані з системи. Ребрами між вершинами є параметр події за допомогою яких між подіями створюються зв'язок. Граф зображення взаємозв’язків є неорієнтованим, наприклад є подія А та подія Б, А є пов’язаною за певним параметром з подією Б, тоді і подія Б буде пов’язаною з подією А.

Методика визначення взаємозв’язків зводиться до послідовності операцій:

- Побудова вершини графу, в якості яких виступають певна подія.
- Побудова загального графу, побудувати ребра між вершинами, зв’язки між вершинами будуються як явні так і не явні.
- Побудова шляху між кожною з вершин за допомогою алгоритму А *, якщо зв'язок між двома подіями є, то ребро стає явним, якщо зв’язку знайти не вдалося то ребра між цими двома вершинами немає.

Приклад побудова загального графу для вибраної події показаний на малюнку (Рисунок 3.1), є події А, Б, С, Д, пунктиром зображені не явні взаємозв’язки, а суцільною лінією явні зв’язки.

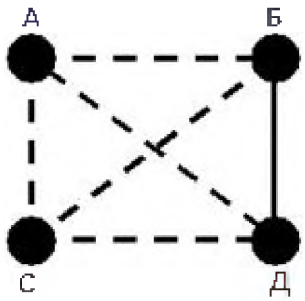

Рисунок 3.1 - загальний граф

Не явний взаємозв’язок між даними, це можливий зв'язок між ними, явним він стане після того як достовірно буде відомо що зв'язок існує між цими даними. Індивідуальний граф взаємозв’язків для події – це граф в якому зображено лише явні взаємозв’язки які отримали за допомогою алгоритму А* між даними (Рисунок 3.2).

Це лише декілька варіантів графу взаємозв’язків для події, вони можуть бути іншими, це залежить від параметрів подій А, Б, С, Д.

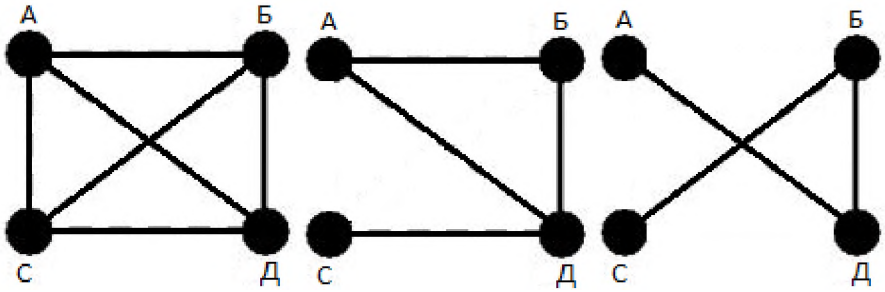

Рисунок 3.2 - Індивідуальний граф

А * Алгоритм пошуку - один з найкращих і найпопулярніших методів, що використовуються для пошуку шляхів та переміщення графів. А * покроково переглядає всі шляхи, що ведуть від початкової вершини в кінцеву, поки не знайде мінімальний. Як і всі поінформовані алгоритми пошуку, він переглядає спочатку ті маршрути, які «здаються» ведуть до мети. Від жодного алгоритму, який теж є алгоритмом пошуку по першому найкращому збігу, його відрізняє те, що при виборі вершини він враховує, крім іншого, *весь* пройдений до неї шлях. Складова $g\ (x)$ - це вартість шляху від початкової вершини, а не від попередньої, як в жадібному алгоритмі.

На початку роботи проглядаються вузли, суміжні з початковим; вибирається той з них, який має мінімальне значення $f\ (x)$, після чого цей вузол розкривається. На кожному етапі алгоритм оперує з безліччю шляхів з початкової точки до всіх ще не розкритих (листових) вершин графа - безліччю приватних рішень, - яке розміщується в черзі з пріоритетом. Пріоритет шляху визначається за значенням $f\ (x) = g\ (x) + h\ (x)$. Алгоритм продовжує свою роботу до тих пір, поки значення $f\ (x)$цільової вершини не виявиться меншим, ніж будь-яке значення в черзі, або поки все дерево не буде переглянуто. З безлічі рішень вибирається рішення з найменшою вартістю.

Чим менше евристика $h\ (x)$, тим більше пріоритет, тому для реалізації черги можна використовувати сортувальні дерева.

Безліч переглянутих вершин зберігається в closed, а які потребують розгляду шляху - в черзі з пріоритетом open. Пріоритет шляху обчислюється за допомогою функції f (x) всередині реалізації черги з пріоритетом.

Приклад реалізації методу псевдокодом:

- Q - безліч вершин, які потрібно розглянути,
- U - безліч розглянутих вершин,
- f[X] - значення евристичної функції "відстань + вартість" для вершини x,
- g[X]- вартість шляху від початкової вершини до x,
- h (x) - евристична оцінка відстані від вершини x до кінцевої вершини.

На кожному етапі роботи алгоритму з безлічі Q вибирається вершина з найменшим значенням евристичної функції і проглядаються її сусіди. Для кожного з сусідів оновлюючи відстань, значення евристичних функції і він додається в безліч Q. Приклад коду:

```
bool A*(start, goal):
   U = ∅
   Q = ∅
   Q.push(start)
   g[start] = 0
   f[start] = g[start] + h(start)
   while Q.size() != 0
      current = вершина з Q з мінімальним значенням f
      if current == goal
         return true      // знайшли шлях до потрібної вершини
      Q.remove(current)
      U.push(current)
      for v : смежные с current вершины
         tentativeScore = g[current] + d(current, v)   // d(current, v) — вартість шляху між current і v if v ∈ U
         if v∈Uv∈U and tentativeScore >= g[v]
            continue
         if v∉Uv∉U or tentativeScore < g[v]
            parent[v] = current
            g[v] = tentativeScore
            f[v] = g[v] + h(v)
            if v∉Qv∉Q
               Q.push(v)
   return false
```

Поведінка алгоритму сильно залежить від того, яка евристика використовується. У свою чергу, вибір евристики залежить від постановки завдання. Часто A * використовується для моделювання переміщення по поверхні, покритої координатної сіткою.

Точна евристика - ми можемо знайти точні значення h, але це зазвичай дуже багато часу. Нижче наведено деякі методи розрахунку точного значення h.

- Перед обчисленням відстані між кожною парою комірок перед запуском алгоритму пошуку A *.
- Якщо немає заблокованих осередків / перешкод, то ми можемо просто знайти точне значення h без будь-яких попередніх обчислень за допомогою формули відстані / евклідової відстані.

В математиці, то евклідова відстань є звичайною прямій лінії відстань між двома точками в евклідовому просторі [13]. Евклідова відстань між точками p і q є довжина відрізка поєднуючи їх ($\overline{pq}$). У декартових координатах, якщо p = (p_1, p_2,..., p_n) і q = (q_1, q_2,..., q_n) є двома точками в евклідовому *n*- простору, то відстань (d) від p до q, або від q до p дається формулою Піфагора (3.1):

$$d(\mathbf{p},\mathbf{q}) = d(\mathbf{q},\mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2} = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}. \quad (3.1)$$

Положення точки в евклідовому *n*- просторі є евклідовим вектором. Таким чином, p і q можуть бути представлені як евклідові вектори, починаючи від походження простору (початкової точки) з їх кінчиками (кінцевими точками), що закінчуються двома точками. Евклидова норма, або евклидова довжина, або величина вектора мір довжини вектора (3.2):

$$\|\mathbf{p}\| = \sqrt{p_1^2 + p_2^2 + \cdots + p_n^2} = \sqrt{\mathbf{p} \cdot \mathbf{p}}, \quad (3.2)$$

Характеризуючи вектор як спрямований відрізок лінії від походження евклідового простору (векторний хвіст) до точки в цьому просторі (кінцева векторна), її довжина фактично є відстанню від її хвоста до її кінчика. Евклідова норма вектора вважається просто евклідовою відстані між її хвостом і кінчиком.

Співвідношення між точками p і q може мати відношення до напрямку (наприклад, від p до q), тому, коли це відбувається, це співвідношення сама може бути представлена вектором, заданим (3.3):

$$\mathbf{q} - \mathbf{p} = (q_1 - p_1, q_2 - p_2, \cdots, q_n - p_n). \tag{3.3}$$

У двох або тривимірному просторі ($n = 2, 3$) це може бути візуально представлено як стріла від p до q. У будь-якому просторі його можна розглядати як положення q відносно p. Його також можна назвати вектором зміщення, якщо p та q представляють два положення деякої рухомий точки. Евклідова відстань між p і q - це лише евклідова довжина цього вектора зміщення (3.4):

$$\|\mathbf{q} - \mathbf{p}\| = \sqrt{(\mathbf{q} - \mathbf{p}) \cdot (\mathbf{q} - \mathbf{p})}. \tag{3.4}$$

що еквівалентно рівнянню 1 (3.5), а також:

$$\|\mathbf{q} - \mathbf{p}\| = \sqrt{\|\mathbf{p}\|^2 + \|\mathbf{q}\|^2 - 2\mathbf{p} \cdot \mathbf{q}}. \tag{3.5}$$

Наближення евристики - Існує, як правило, три наближенні евристики для обчислення h:

- **Manhattan Distance -** Це не що інше, як сума абсолютних значень різниць у координатах x та y цілі та координати x та y поточної комірки відповідно, тобто h = abs (current_cell.x - goal.x) +abs (current_cell.y - goal.y). Цей підхід використувується лише коли нам дозволяється рухатися лише в чотирьох напрямках (праворуч, ліворуч, зверху, знизу)
- **Diagonal Distance** - це ніщо інше, як максимум абсолютних значень різниць у координатах x та y цілі та координати x та y поточної комірки відповідно, тобто h = max {abs (current_cell.x - goal.x), abs (current_cell.y -

goal.y)}ми використовуємо цей підхід коли нам дозволено рухатися лише у восьми напрямках (подібно до руху короля в шахи) діагональна дистанційна евристика показана на малюнку нижче (Рисунок 3.3), припустимо, червоне місце як джерело клітини і зелене місце як клітина-мішень.

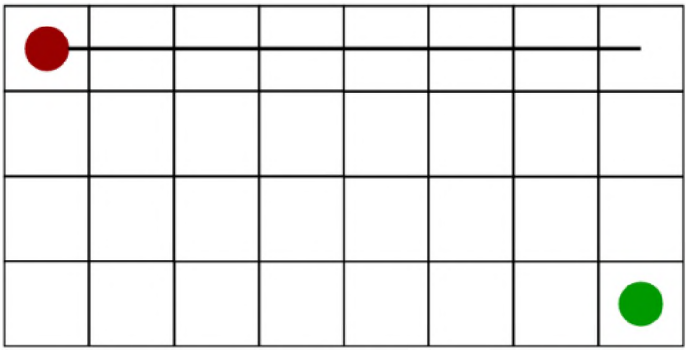

Рисунок 3.3 - Діагональна дистанційна

- **Euclidean Distance -** Як видно з його назви, це не що інше, як відстань між поточною клітиною та цільовою клітиною за допомогою формули відстані h = sqrt ((current_cell.x - goal.x) 2 + (current_cell.y - goal.y) 2) Використовуємо коли нам дозволено рухатися в будь-яких напрямках. Euclidean Distance евристика показана на малюнку нижче (Рисунок 3.4) припустимо, червоне місце як джерело клітини і зелене місце як клітинку-мішень.

-

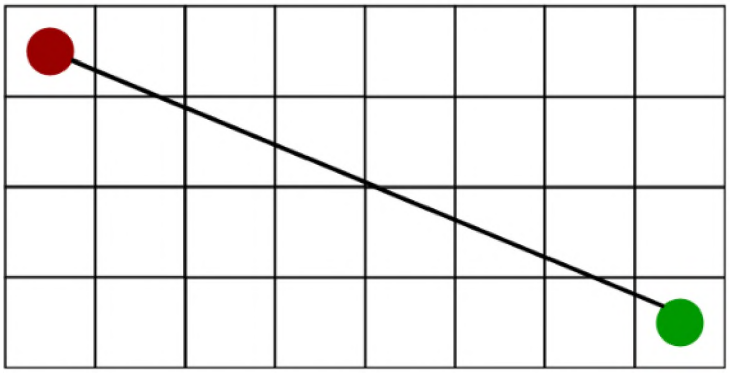

Рисунок 3.4 - Euclidean Distance

Давайте розглянимо приклад взаємозв'язків на основі даних інформаційної системи. Загальний граф для подій буде виглядати приблизно так (Рисунок 3.5):

Рисунок 3.5 - Загальний граф для подій

Пунктиром зображені не явні взаємозв'язки, а суцільною лінією явні зв'язки.

Наприклад якщо викликаються дані про облікові записи то точно буде детальна інформація про певний обліковий запис, інформація про ПК в якому знаходяться ці облікові записи.

Розглянимо приклад графу відносно певної події із списку встановлених програм. Перед побудовою індивідуального графу взаємозв'язків для вибраної події перевіряємо з якими даними ця подія пов'язана. Наприклад візьмемо програму для аналізу встановлену програму "Google Chrome", Вона на даний момент відображається в запущених процесах, тож ми можемо переглянути детальну інформацію про нього коли був запущен, ім'я процесу, та реальну директорія з якої запускаються процес. Це є корисним адже зловмисник іноді замінюють ім'я процесу на відомі або системні назви процесів а на справді запускається зловмисна програма під цим ім'ям. Далі перевіряємо який

користувач запустив цей процес і детальну інформацію про обліковий запис. Логів входів користувача та встановлених оновлень наприклад може не знайти, адже зловмисник або його програма видалила цю інформацію щоб ускладнити роботу спеціалістів. З рештою перевіряємо веб ресурси, деякі події в кожній з категорій використовували Google Chrome для доступу в інтернет. Також в планувальнику подій знайдено подія яка при включенні системи запускає Google Chrome, а в ньому відкриває сайт зловмисник, це може бути або спам або програма для скритного майнінгу. Переглянувши інформацію про це посилання в веб ресурсам можна виявити що вже були переходи за цим посиланням, та з цього сайту йде завантаження певного файл. Індивідуальний граф взаємозв'язків для вибраної події та змодельованої ситуації буде приблизно такий (Рисунок 3.6):

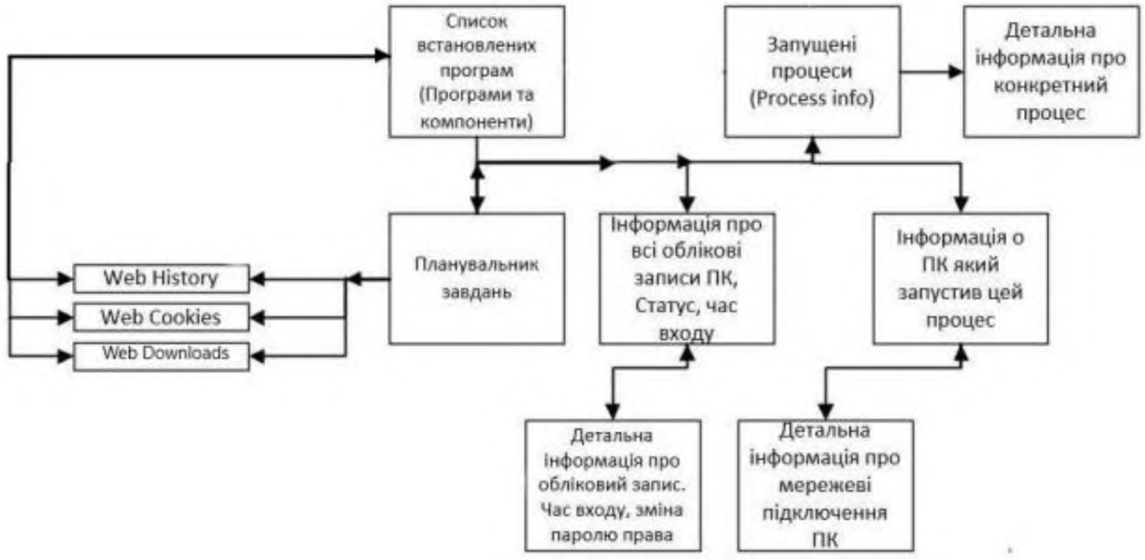

Рисунок 3.6 - Індивідуальний граф взаємозв'язків

Такий підхід на мою думку дає більш точне зображення подій які відбувалися в інформаційній системі. Спеціаліст може вибрати певну підозрілу подію та одразу побачити усі взаємопов'язані данні з нею і побудувати послідовність подій для них.

3.2 Опис використаних засобів

3.2.1 Операційна система

У якості операційної системи (ОС) обрано OS Windows 10. Windows 10 – операційна система від компанії Microsoft для персональних комп'ютерів, ноутбуків, планшетів, лептопів-трансформерів і смартфонів. Даний вибір обумовлено тим, що було проведення зчитування даних на OS Windows та OS Linux, але результати на OS Windows був отриманий більш наочним тому я вибрав її для подальшої роботи. Існують багато версій операційної системи Windows, одні з найвідоміших це Windows XP, Windows Vista, Windows 7, Windows 8, Windows 10.

Я обрав саме OS Windows 10, так як на сьогоднішній час це найбільш сучасна, має більший функціонал та можливості порівняно з попередніми версіями Windows. Система покликана стати єдиною для різних пристроїв, таких як персональні комп'ютери, планшети, смартфони, консолі Xbox One і ін. Доступна єдина платформа розробки і єдиний магазин універсальних програм, сумісних з усіма підтримуваними пристроями. Windows 10 поставляється в якості послуги з випуском оновлень протягом усього циклу підтримки. Протягом першого року після виходу системи користувачі могли безкоштовно оновитися до Windows 10 на пристроях під управлінням ліцензійних копій Windows 7, Windows 8.1 і Windows Phone 8.1. Серед значимих нововведень - голосова помічниця Кортан, можливість створення і перемикання декількох робочих столів. Ще одна особлива функція в Windows 10 — якщо пристрій під'єднаний до Інтернету і ви під'єднали до комп'ютера незнайомий пристрій для комп'ютера або ОС, то не потрібно буде шукати самому драйвера під цей пристрій, Windows його знайде, завантажить та повторно підключить пристрій до комп'ютера.

Окрім того що Windows є сучасною операційною системою, вона більш вразлива до нападів кіберзлодії та вірусів. Тому я зосередив свою роботу саме на цій операційній системі.

3.2.2 Середовище розробки

Інтегроване середовище розробки являє собою програмний додаток, який надає комплексні можливості для програмістів для розробки програмного забезпечення. Середовище розробки включає в себе:

- текстовий редактор;
- компілятор і / або інтерпретатор;
- засоби автоматизації збирання;
- відладчик.

Більшість сучасних IDE мають інтелектуальне завершення коду. Деякі IDE, такі як NetBeans та Eclipse, містять компілятор, інтерпретатор; інші, такі як SharpDevelop і Lazarus, не роблять. Іноді IDE містить також засоби для інтеграції з системами управління версіями і різноманітні інструменти для спрощення конструювання графічного інтерфейсу користувача. Багато сучасні середовища розробки також включають браузер класів, інспектор об'єктів і діаграму ієрархії класів - для використання при об'єктно-орієнтованої розробки ПЗ. IDE зазвичай призначені для декількох мов програмування - такі як IntelliJ IDEA, NetBeans, Eclipse, Qt Creator, Geany, Embarcadero RAD Studio, Code Blocks, Xcode або Microsoft Visual Studio, але є і IDE для одного певного мови програмування - як, наприклад, Visual Basic, Delphi, Dev-C ++.

У якості середовища розробки обрано Microsoft Visual Studio 2017. Visual Studio є дійсно потужним середовищем розробки програмного забезпечення і незаперечно є одним з провідних продуктів на ринку програмних засобів, які використовуються для розробки програмного забезпечення. На мій погляд даний продукт найбільш підходить для великомасштабних проектів, так як кожен продукт забезпечений дуже зручною інформаційною мережею MSDN, яка забезпечує додатковими бібліотеками для розробників і техподдержкой клієнтів. Тому при виборі серодовища розробки я зупинився на Microsoft Visual Studio 2017, так як вона найбільш підходить до вимог моєї програми.

Microsoft Visual Studio представляє собою інтегроване середовище розробки (IDE) від Microsoft. Він використовується для розробки комп'ютерних програм для Microsoft Windows, а також веб-сайтів, веб-додатків, веб-сервісів і мобільних додатків. Visual Studio використовує платформи розробки програмного забезпечення Microsoft, такі як Windows API, Windows Forms, Windows Presentation Foundation, Windows Store і Microsoft Silverlight. Він може виробляти як машинний код і керований код. Visual Studio включає в себе редактор коду, що підтримує IntelliSense (компонент код завершення), а також рефакторинг коду. Вбудований відладчик працює і як відладчик вихідного рівня і відладчик на рівні машини. Інші вбудовані засоби включають в себе код профайлер, формує конструктор для створення додатків з графічним інтерфейсом, веб-дизайнер, клас дизайнера і конструктора схеми бази даних. Він приймає плагінів, які розширюють функціональність майже на кожному рівні, включаючи додавання підтримки систем управління версіями (наприклад, Subversion) і додавання нових наборів інструментів, таких як редактори та візуальні дизайнери для предметно-орієнтованих мов або наборів інструментів для інших аспектів життєвого циклу розробки програмного забезпечення (як клієнт Team Foundation Server: Team Explorer). Visual Studio підтримує 36 різних мов програмування і дозволяє редактор коду і відладчик для підтримки (у різному ступені) практично на будь-якій мові програмування, за умови обслуговування конкретного мови існує. Вбудовані мови включають C, C ++ і C ++, VB.NET, C #, F #. Підтримка інших мов, таких як Python, Рубін, Node.js і M серед інших можна за допомогою мовних служб, встановлених окремо. Він також підтримує XML / XSLT, HTML / XHTML, JavaScript і CSS. Java (і J #) були підтримані в минулому.

Різні програми багатовіконного середовища часто виконують однакові дії, наприклад, хрестик в правому верхньому куті вікна, який закриває його, малюється більшістю програм однаково. Марнотратно було б, щоб кожна із цих програм мала відповідну функцію — це роздувало б їхні розміри. Тому, розумно, щоб такі функції поступали в спільне користування. Для цього служать бібліотеки з динамічним зв'язуванням. Відповідні функції завантажуються в пам'ять

комп'ютера не з файлу програми, а із спеціального файлу, вже при виконанні. Насправді, операційна система не завантажує їх повторно. Якщо програма при запуску вимагає завантаження динамічної бібліотеки, то операційна система перевіряє, чи така бібліотека вже є в пам'яті. Якщо вона є, то операційна система збільшує лічильник клієнтів для динамічної бібліотеки на одиницю. При завершенні роботи, програма повідомляє операційну систему про необхідність вивантажити динамічну бібліотеку. При цьому операційна система зменшує лічильник клієнтів на одиницю. Якщо після такого зменшення кількість клієнтів стає рівною нулю, то тоді динамічна бібліотека справді вивантажується із пам'яті комп'ютера.

Мережа Розробників Майкрософт це частина компанії Майкрософт, яка відповідає за співробітництво компанії з громадою розробників програмних продуктів на базі власних розробок. Підрозділ працює як інформаційний сервіс для розробників програмного забезпечення. Основна увага, останнім часом, приділяється платформі Microsoft.NET, але присутні і статті, що охоплюють такі області як практика програмування і шаблони проектування.

3.2.3 Мова програмування

Я використав мову C#. C# — об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи.NET. Синтаксис C# близький до C++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато що від своїх попередників — мов C++, Delphi, Модула і Smalltalk — C#, спираючись на практику їхнього використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад множинне спадкування класів (на відміну від C++). Відзначимо наступні важливі фактори:

- C# створювався і розвивається паралельно з каркасом Framework.Net і в повній мірі враховує всі його можливості;
- C# є повністю об'єктно-орієнтованою мовою;
- C# є потужним об'єктним мовою з можливостями успадкування та універсалізації;
- C# є спадкоємцем мови C++. Загальний синтаксис, загальні оператори мови полегшують перехід від мови C++ до C#;
- Зберігши основні риси свого батька, мова стала простіше і надійніше;
- Завдяки каркасу рамки.Net, який став надбудовою над операційною системою, програмісти C# отримують переваги роботи з віртуальною машиною;
- Framework.Net підтримує різноманітність типів додатків на C#;
- Реалізація, що поєднує побудова надійного і ефект.

Незважаючи на те, що синтаксис і особливість реалізація успадковала мова програмування C# від «прабатьків» (C++, Java), можливості цього мови програмування не обмежуються ними. До принципово важливих рішень, які були реалізовані, можна віднести наступні:

- компонентно-орієнтований підхід до програмування;
- властивості як засіб інкапсуляції даних);
- обробка подій (маються розширення, в тому числі в частині обробки винятків, зокрема, оператор спробувати);
- уніфікована система типізації (відповідає ідеології Microsoft.NET в цілому);
- делегати (делегат - розвиток покажчика на функцію в мовах C і C ++);
- індексатори (indexer - оператори індекс для звернення до елементам клас-контейнер);
- перевантажені оператори;
- оператор foreach (обробка всіх елементів класів-колекцій);
- механізми boxing и unboxing для перетворення типів;

- атрибути (засіб оперування метаданих в COM-модель);
- прямокутні масиви (набір елементів з доступом за номером індекс і однакову кількість стовпців і рядків).

Основні переваги мови C#:

- Мова програмування C# об'єктно-орієнтована;
- Компонентно-орієнтований підхід до програмування, сприяє менша машинно-архітектурна залежність результуюча програмний код, гнучкість, переносимість і легкості повторне використання (фрагменти) програми;
- Орієнтація на безпеку коду (в порівнянні з C і C ++);
- Уніфікована система типізації;
- Розширена підтримка подієво-орієнтована програмування.

На даний момент, C# успішно конкурує з Java і C ++ по популярності. Для початку перерахуємо подібності мов програмування C# і Java. Обидві мови об'єктно-орієнтовані і припускають єдиність успадкування. Так само особливості, які роблять схожими мови програмування C# і Java, є механізми інтерфейсів, обробки виняткових ситуацій, нитки (потоки). Обидва мови має строгу типізацію і динамічне завантаження коду при виконання програми.

Від мови програмування C++, C # мовою успадковані механізми: «перевантажений» оператори, небезпечний арифметичний операції з плаваючою точкою і безліч інших особливостей синтаксису.

3.2.4 Використаний API

У комп'ютерному програмуванні, інтерфейс прикладного програмування (API) являє собою набір визначень підпрограм, протоколів та інструментів для створення прикладного програмного забезпечення. У загальних рисах, це набір чітко визначених методів зв'язку між різними компонентами програмного забезпечення. Гарний API спрощує розробку комп'ютерної програми, надаючи всі

будівельні блоки, які потім разом узяті програмістом. API може бути для веб-системи, операційної системи, системи баз даних, комп'ютерного обладнання або програмного забезпечення бібліотеки. Специфікація API може приймати різні форми, але часто включає в себе специфікацію для процедур, структур даних, класів об'єктів, змінних. Одні з прикладів API :

- Windows API
- OpenGL
- DirectX
- GDI
- MARF

У моєї програми був використаний API Windows Forms, для забезпечення візуальної демонстрації роботи збору волатильних даних з операційної системи, побудови графу взаємозалежності даних, та відтворення послідовності подій в ОС.

Windows Forms - інтерфейс програмування додатків (API), що відповідає за графічний інтерфейс користувача і є частиною Microsoft.NET Framework. Даний інтерфейс спрощує доступ до елементів інтерфейсу Microsoft Windows за рахунок створення обгортки для існуючого Win32 API в керованому коді. Причому керований код - класи, що реалізують API для Windows Forms, що не залежать від мови розробки. Тобто програміст однаково може використовувати Windows Forms як при написанні ПЗ на C #, C ++, так і на VB.Net, J #. Додаток Windows Forms є Об'єктно-орієнтований додаток, підтримуване Microsoft.NET Framework. На відміну від пакетних програм, велика частина часу витрачається на очікуванні від користувача будь-якого дії, як, наприклад, введення тексту в текстовому полі або кліки мишки по кнопці.

У Windows Forms форма - це візуальна поверхню, на якій виводиться інформація для користувача. Зазвичай додаток Windows Forms будується шляхом приміщення елементів управління на форму і написання коду для реагування на дії користувача, такі як клацання миші або натискання клавіш. Елемент

управління - це окремий елемент призначеного для користувача інтерфейсу, призначений для відображення або введення даних.

Windows Forms включає широкий набір елементів управління, які можна додавати на форми: текстові поля, кнопки, списки, що розкриваються, перемикачі та навіть веб-сторінки. Якщо існуючий елемент управління не задовольняє потребам, в Windows Forms можна створювати власні елементи управління за допомогою класу UserControl.

Якщо потрібно створити свої власні елементи призначеного для користувача інтерфейсу, простір імен System.Drawing містить широкий набір класів, необхідних для відтворення ліній, кіл та інших фігур безпосередньо на формі.

Microsoft.NET Framework — програмна технологія, запропонована фірмою Microsoft як платформа для створення як звичайних програм, так і веб-застосунків. Багато в чому є продовженням ідей та принципів, покладених в технологію Java. Одною з ідей.NET є сумісність служб, написаних різними мовами. Хоча ця можливість рекламується Microsoft як перевага.NET, платформа Java має таку саму можливість. Кожна бібліотека (збірка) в.NET має свідчення про свою версію, що дозволяє усунути можливі конфлікти між різними версіями збірок. Мови програмування в.NET є вбудовані та ті які що постачаються окремо. Вбудовані (постачаються разом з.NET Framework): C#, J#. VB.NET, JScript.NET, C++/CLI.

3.2.5 Програма збору не волатильних даних

Не волатильні дані, це дані які зберігаются в незалежній пам'яті (NVM). Незалежній пам'ять - це тип пам'яті комп'ютера, який може витягувати збережену інформацію навіть після циклічного включення живлення. Навпаки, енергозалежна пам'ять потребує постійної потужності для збереження даних. Приклади незалежній пам'яті включають в себе постійний запам'ятовуючий

пристрій, флеш-пам'ять, сегнетоелектричних ОЗП, більшість типів магнітних запам'ятовуючих пристроїв (наприклад, жорсткі диски, твердотільні накопичувачі, гнучкі диски і магнітну стрічку), оптичні диски, і ранні комп'ютерні методи зберігання, такі як паперова стрічка і перфокарт.

Незалежна пам'ять може бути класифікована як традиційне незалежне дисковий сховище або сховище в енергонезалежних мікросхемах пам'яті (флеш-пам'ять) - EEPROM, SSD, NAND і т.д.

Тимчасові (temp) файли: temp файл створюється програмою, коли вона не може виділити достатньо пам'яті для своїх завдань, або програма працює над великим набором даних. Як правило, файли temp видаляються, коли програма закінчується. Однак деякі програми створюють тимчасові файли і залишають їх позаду.

Реєстри систем. Реєстр - це база даних, яка містить інформацію та настройки апаратного забезпечення системи, операційної системи, переваг користувача та обчислювальних операцій.

Журнали подій: відповідно до встановлених системним адміністратором параметрів, журнали подій записують певні події, надаючи аудиторський слід, який може використовуватися для діагностики проблем або для розслідування підозрілих дій.

Завантажувальні сектори: жорсткі диски у виробничих середовищах, як правило, організовані в декілька розділів. Кожен розділ може мати іншу операційну систему. Завантажувальні сектори містять інструкції щодо завантаження операційних систем.

Кеш веб-браузера: веб-переглядачі дозволяють користувачам локально кешувати вміст веб-сторінок, щоб швидше отримати доступ до часто відвідуваних сайтів. Завантажений вміст залишається на жорсткому диску, доки не буде видалено. Однак, навіть після кешування

видаляється, дані можуть залишатися в нерозподіленому просторі жорсткого диска.

Файли cookie. Веб-сервери взаємодіють із веб-браузерами через файли cookie: невеликі пакети даних, які використовуються для відстеження, перевірки та підтримки конкретної інформації про користувачів. Cookie може мати термін придатності, коли браузер видаляє його.

Cookie-файли без дати закінчення терміну дії видаляються в кінці сеансу користувача. Користувачі можуть також видаляти файли cookie за їх запитом. Однак, як і кеш веб-браузера, дані cookie можуть залишатися в нерозподіленому просторі жорсткого диска після його видалення.

Для зчитування не волатильних даних я використовував програму Autopsy. Autopsy - це платформа цифрової криміналістики та графічного інтерфейсу для Sleuth Kit та інших цифрових криміналістичних інструментів. Вона використовується правоохоронними органами, військовими та корпоративними експертами для розслідування наслідків на комп'ютерах.

Інструмент розроблений з урахуванням цих принципів:

- Extensible - користувач повинен мати можливість додавати нові функції, створюючи плагіни, які можуть аналізувати все або частину базового джерела даних.
- Centralised - інструмент повинен запропонувати стандартний і послідовний механізм доступу до всіх функцій та модулів.
- Ease of Use - браузер Autopsy має запропонувати інструменти, щоб користувачам було простіше повторити свої кроки без надмірної реконфігурації.
- Multiple Users - цей інструмент повинен бути використаний одним дослідником або координувати роботу команди.

Основні можливості Autopsy:

- Аналіз активності за часом: Показ системних подій в графічному інтерфейсі для допомоги в ідентифікації активності.
- Пошук за ключовими словами: Витяг тексту і модулі індексного пошуку дають вам можливість знайти файли, які згадують специфічні терміни і здійснювати пошук по паттернам регулярних виразів.

- Веб артефакти: Витяг веб активності з популярних браузерів для допомоги в ідентифікації користувача активності.
- Аналіз реєстру: Використовується RegRipper для ідентифікації доступу до останніх документів і USB пристроїв.
- Аналіз файлів LNK: Визначає ярлики і відкриті документи.
- Аналіз електронної пошти: Розбір повідомлень в форматі MBOX, таким як Thunderbird.
- EXIF: Витягує інформацію про геолокації і камері з файлів JPEG.
- Сортування за типами файлів: Угруповання файлів по їх типу для пошуку всіх зображень або документів.
- Відтворення медіа: Переглядайте відео та зображень в додатку, зовнішній переглядач не потрібно.
- Перегляд мініатюр: Показує мініатюри зображень для допомоги в швидкому огляді картинок.
- Надійний аналіз файлової системи: Підтримка популярних файлових систем, включаючи NTFS, Yaffs2 і UFS, HFS +, ISO9660 (CD-ROM), Ext2 / Ext3 / Ext4, FAT12 / FAT16 / FAT32 / ExFAT з The Sleuth Kit.
- Фільтрація файлів по Хешам: відфільтровування добре відомих файлів з використанням NSRL і позначка поганих файлів, використовуючи призначені для користувача наборі хешів в форматах HashKeeper, md5sum і EnCase.
- Теги: Позначте файли тегами, з довільними іменами тегів, такими як «закладки», «підозрілі» та додавайте коментарі.
- Витяг рядків Unicode: Отримуйте рядки з НЕ розподілених областей і невідомих типів файлів на багатьох мовах (арабською, китайською, японською і т. Д.).
- Визначення типу файлу на основі сигнатур і виявлення невідповідності розширення файлу його вмісту.
- Модуль цікавих файлів позначить файли і папки, грунтуючись на імені і шляхи.

- Підтримка Android: Витяг даних з SMS, журналу дзвінків, контактів, Tango, Words with Friends та інших.

Під час збору не волотильних даних є певні принципи та правели яких потрібно дотримуватися:

- Розробити та впровадити ланцюжок опіки, який є процесом відстеження зібраної інформації та збереження цілісності інформації. За наявності документально підтвердженого ланцюжка опіки обов'язково, якщо зібрані дані будуть використані в судовому процесі. Окрім юридичних вимог для збору доказів, найкращим є практика проведення всіх розслідувань з порушенням, використовуючи стандартну методику збору даних.
- Зробіть біт-біту копію (біт-потоку) жорсткого диска системи, який фіксує кожен біт на жорсткому диску, у тому числі слабкий простір, нерозподілений простір та файл підкачки.
- Обчисліть хеш-значення бітів-потокового зображення та інших досліджуваних файлів. Для хеш-даних використовується перетворення існуючих даних у невеликий потік символів, який слугує відбитком даних. Хайшинг диски та файли забезпечують їх цілісність і автентичність. Ці характеристики повинні зберігатися, якщо докази повинні використовуватися в судовому розгляді.
- Не працюйте над оригінальними цифровими доказами. Це незрозуміло, але його можна не помітити. Після створення подвійного дубліката підозрілого диска, оригінальні диски повинні бути доступні якомога менше. Будь-яка розслідувальна робота повинна виконуватися на зображенні бітового потоку.
- Проведіть цифровий диктофон для запису розмов з персоналом, який бере участь у розслідуванні. Дослідження порушень часто включають вихор бесід, декларацій та інших тверджень, які можуть бути корисними в ході розслідування. Використання цифрового диктофона економить аналітиків від необхідності відкликати всі мінуси, що є поверхнями під час розслідування. Архівувати / організовувати / асоціювати всі цифрові

голосові файли поряд з іншими доказами, зібраними під час розслідування.

Для зчитування цих даних з жорсткого диску я використав програму Autopsy.

Autopsy є графічним інтерфейсом для інструментів аналізу цифрових досліджень командного рядка в наборі Sleuth Kit. Разом вони можуть аналізувати диски та файлові системи Windows і UNIX (NTFS, FAT, UFS1 / 2, Ext2 / 3). Було зроблено копію жорсткого диска та зчитування не волатильних даних з систем Windows та Linux. Результати вийшли приблизно однакові але аналіз диску з операційної системи Linux, видав набагато менше даних для подальшої обробки. Це можна пояснити тим що ОС Linux я встановив відносно не давно, тому я продовжив далі працювати саме з даними які зібрав з ОС Windows.

Пошук за ключовими словами - в результаті Autopsy використовує потужний засіб індексації тексту Apache SOLR для швидкого та надійного пошуку ключових слів. Заздалегідь визначені списки ключових слів і регулярних виразів можуть бути налаштовані на запуск під час зйомки зображення. За промовчанням в Autopsy включаються пошуки регулярного виразу для:

- Адреси електронної пошти
- Телефонні номери
- IP-адреси
- URL-адреси

Крім того, спеціальні пошукові запити на ключові слова можуть бути запущені безпосередньо з панелі вмісту під час розслідування після (або навіть протягом) знімання зображення диска. Кілька одночасних пошуків за ключовими словами можна запустити з пошуковим індексом.

Замість того, щоб шукати вихідні дані, пошук за ключовим словом у Autopsy виконується на виході модулів вилучення тексту. Autopsy використовує Tika та інші бібліотеки для вилучення тексту з HTML, Microsoft Office, PDF, RTF тощо. Цей підхід ефективніший при пошуку тексту, ніж пошук на небанківських файлах PDF та файлах docx, за допомогою яких дані стискуються.

Будь-які результати пошуку, які увімкнуті (регулярний вираз або списки визначених користувачем), відображатимуться в вузлі "Класичні звернення" у навігаційному дереві Autopsy. Завдяки Apache SOLR всі файли Autopsy ідентифікують те, що текстовий вміст у них буде індексуватися для пошуку за допомогою попередньо визначених списків регулярних виразів, списків ключових слів або спеціальних запитів.

Відтворення мультимедіа – зчитано EXIF з фотографій та дивіться відеоролики. EXIF це стандарт, що визначає формат описання допоміжної метаінформації для файлів зображень, звуку, і який використовується цифровими камерами (в тому числі і тими що у смартфонах), сканерах і іншими системами, що обробляють звукові файли та файли зображень записані цифровими камерами. Специфікація використовує наступні формати файлів, в які додаються спеціальні теги метаданих: JPEG і дискретне косинусне перетворення (DCT) для файлів зображень зі стисненням даних, TIFF (RGB або YCbCr) для нестиснутих файлів зображень і RIFF WAV для аудіо файлів. Він не використовується в JPEG 2000, PNG, або GIF.

Аналіз реєстру - використовує RegRipper для визначення останніх документів і USB-пристроїв. RegRipper - це криміналістичне програмне забезпечення, розроблене, з відкритим вихідним кодом. RegRipper, написаний на Perl і є інструментом вилучення даних реєстру Windows. RegRipper може бути налаштований відповідно до потреб експерта за допомогою наявних плагінів або користувачів, що пишуть плагіни відповідно до конкретних потреб.

Відновлення видалених файлів з нерозподіленого простору за допомогою PhotoRec. PhotoRec - це програма для відновлення файлів, призначена для відновлення втрачених файлів, включаючи відео, документи та архіви з жорстких дисків, компакт-дисків та втрачених зображень (таким чином, ім'я відновлення фотографій) із пам'яті цифрової камери. PhotoRec ігнорує файлову систему та йде за основними даними, тому вона буде працювати, навіть якщо файлова система вашого носія була сильно пошкоджена або переформатована.

Аналіз веб-артефактів - Autopsy призначений для пошуку загальних веб-артефактів у сучасних основних браузерах, зокрема:

- Firefox
- Chrome
- Internet Explorer

Autopsy витягує наступну інформацію та публікує її на дошці:

- Закладки
- Cookies
- Історія
- Завантаження
- Пошукові запити

Щоб полегшити пошук даних, результати всіх веб-переглядачів об'єднуються. Отже, якщо ви хочете побачити історію користувача, перейдіть до вузла історії. Вам не потрібно переходити через папки для різних веб-переглядачів, перш ніж ви зможете знайти історію.

Всі вони класифікуються та відображаються автоматично у вікні дерева Autopsy під вузлом "Результати". Ця функція дає досліднику можливість автоматичного швидкого доступу до інформації про рівень роботи програми через веб-переглядачі та може призвести до додаткових пошукових запитів ключових слів і гіпотез дослідження для вивчення.

Після завантаження не волотильних даних вони відображаються у вікні Autopsy у вигляді дерева(Рисунок 3.7).

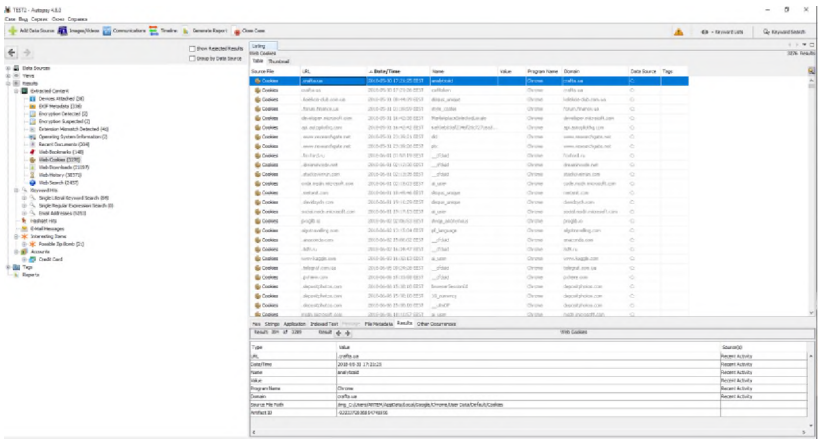

Рисунок 3.7 - Відображення результатів в Autopsy

В нижній частині програми можна переглянути вміст вибраного файлу. Він показує зображення, відео, шістнадцятковий текст, витягнуті рядки, метадані і т.д. Вони активовані, коли ви вибираєте файл у списку файлів над ним (Рисунок 3.8).

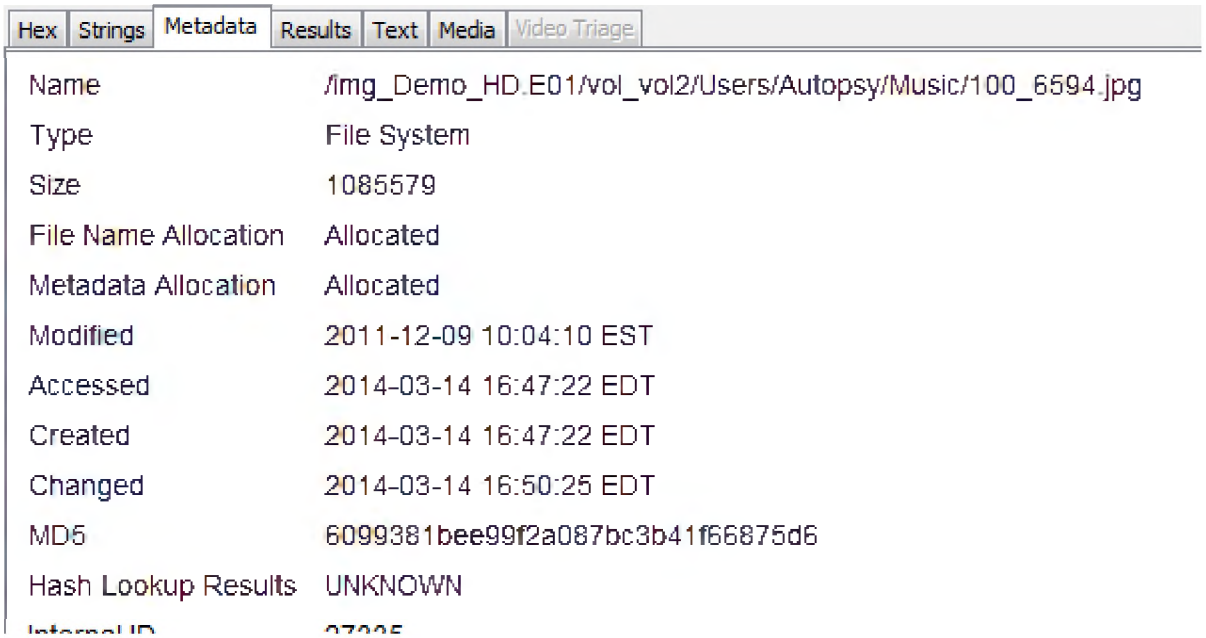

Рисунок 3.8 - Autopsy перегляд вмісту вибраного файлу

Контент-переглядач контекстно-орієнтований, тобто він буде представляти різні погляди на вміст залежно від обраного типу файлу. Наприклад, a.JPG відображатиметься як зображення, текстовий файл відображатиметься як текст, а файл.bin буде відображатися як шістнадцятковий вивід.

Після завантаження даних та їх перегляду є можливість експортувати отримані дані в один із форматів: Excel, HTML, Text, KML, STIX, TSK Body file. Користувач має можливість вибрати один із цих для експорту, та дані які потрібно експортувати залежно від подальших потреб. Я експортував дані у форматі Excel. Кожний тип даних він зберігає як новий лист, що є дуже зручним для мене для подальшої обробки цих даних.

Основні переваги програми Autopsy:

Швидкодія - Autopsy запускає фонові завдання паралельно, використовуючи безліч ядер і виводить результати відразу після їх виявлення. На повне вивчення диска можуть піти годинник, але вже через хвилини, якщо ваші ключові слова

були знайдені в призначеній для користувача домашньої папки, ви про це дізнаєтеся.

Рентабельність - Autopsy безкоштовна. Якщо бюджет урізають, без економічно ефективних цифрових криміналістичних інструментів не обійтися. Autopsy пропонує ті ж основні функції, що й інші інструменти для цифрової криміналістики, а також пропонує інші основні функції, такі як аналіз веб артефактів і аналіз регістра, які відсутні в інших комерційних інструментах.

Складання звітів та експорт результатів – Autopsy має розширену інфраструктуру звітності, яка дозволяє створювати дослідникам додаткові типи звітів. За замовчуванням доступні звіти в файлах HTML, XLS і Body.

3.2.6 База даних

База даних являє собою набір інформації, яка організована таким чином, що її можна легко отримати, управляти і оновлювати.

Дані впорядковані по рядках, стовпцях і таблицями і індексуються для полегшення пошуку релевантної інформації. Дані оновлюються, розширюються і видаляються в міру додавання нової інформації. Бази даних обробляють робочі навантаження для створення та оновлення самих себе, запиту даних, які вони містять, і запуску додатків проти нього.

У моїй роботі я вирішив використати базу даних для більш швидкого зберігання інформації, її виводу та обробки даних. Тому було питання яку з існуючих БД вибрати. Проаналізувавши завдання сформулювали наступні вимоги до сховища:

- Це повинно бути БД яку легко можна застосувати к моєму проекту.
- Швидкість запису, зчитування та модифікація даних.
- Легкість розгортання бази даних

Після ознайомлення було вибрано два варіанти SQLite та LiteDB.

SQLite - класична реляційна база даних. SQLite є ACID- сумісним і реалізує більшість стандартів SQL, використовуючи динамічно і слабо типізований синтаксис SQL, який не гарантує цілісність домену [6]. SQLite є популярним вибором в якості програмно-апаратних засобів бази даних для локального-клієнтського сховища в прикладному програмному забезпеченні, такому як веб-браузери, програмні додатки. Це, можливо, найбільш широко розгорнутий механізм бази даних, оскільки він використовується сьогодні декількома широко поширеними браузерами, операційними системами і вбудованими системами (такими як мобільні телефони) і іншими. SQLite має прив'язки до багатьох мов програмування.

База представлена одним файлом на диску. На цей файл завантажується схема даних, після чого з нею доведеться взаємодіяти засобами SQL. Можна буде створити дві таблиці: одну для властивостей, іншу для контенту, на випадок якщо будуть завдання де знадобиться одне без іншого.

ACID - це набір властивостей, що гарантують надійну роботу транзакцій бази даних: атомарність, узгодженість, ізольованість, довговічність. В контексті баз даних, послідовність операцій з базою даних, яка задовольняє властивостям ACID, можна розглядати як одну логічну операцію над даними. Така послідовність операцій називається транзакцією. Наприклад, переказ коштів з одного банківського рахунку на інший, містить численні операції, але є єдиною транзакцією.

Основні вимоги:

- Атомарність гарантує, що жодна транзакція не буде виконана частково. Будуть або виконані всі операції, що беруть участь у транзакції, або не виконано жодної. Якщо протягом роботи однієї з операцій виникне помилка і операцію буде відхилено, то будуть відхилені також усі інші зміни, здійснені в межах транзакції.
- Відповідно до вимоги узгодженості, система повинна перебувати в узгодженому, несуперечливому стані до початку дії транзакції і по її завершенню. При цьому вона може перебувати в неузгодженому стані

протягом виконання транзакції, проте ця неузгодженість завдяки іншим властивостям — атомарності та ізольованості — не буде видимою за межами транзакції.

- Наприклад, при переведенні коштів з рахунка на рахунок, кошти можуть спочатку зніматись з першого рахунку, після чого нараховуватись на другий. Відповідно, після зняття коштів, але до їх нарахування система перебуває в неузгодженому стані — коштів немає на жодному з рахунків. Але після завершення транзакції повна сума перебуватиме на другому (або першому у випадку скасування транзакції) рахунку.
- Ізольованість означає, що жодні проміжні зміни не будуть видимі за межами транзакції аж до її завершення. Питання ізоляції стає актуальним при одночасній роботі багатьох транзакцій з одними й тими самими даними. Згідно з цією вимогою, якщо дві транзакції намагатимуться змінити одні й ті самі дані, то одну з них буде відхилено або призупинено до завершення другої.
- Довговічність гарантує, що незалежно від інших проблем після відновлення працездатності системи результати завершених транзакцій будуть збережені. Іншими словами, якщо користувач отримав повідомлення про успішне завершення транзакції, то він може бути впевнений, що дані будуть збережені та відновлені у випадку збоїв [17].

LiteDB - нереляційних база, є безсерверной базою даних, що поставляється в одній DLL (менш 350 кб), повністю написаної в керованому коді.NET C # (сумісному з.NET 3.5, 4.x, NETStandard 1.3 і 2.0). LiteDB був натхненний базою даних MongoDB, і його API дуже схожий на офіційний.NET-інтерфейс MongoDB. LiteDB встановлюється через NuGet або просто скопіюйте DLL в папку проекту bin. Як і в SQLite база представлена одним файлом. Зацікавила простота використання: всього лише потрібно віддати бібліотеці об'єкт, а сериализацию вона вже бере на себе.

Було проведено тесту продуктивності між SQLite та LiteDB щоб визначити яку з баз даних використовувати в роботі. Загальна ідея була така: порівняти як

довго буде записуватися в базу багато маленьких файлів, середня кількість середніх за розміром файлів і трохи дуже великих файлів. Варіант із середніми і великими файлами найбільш близький до реального, а маленькі файли це прикордонний випадок, який теж потрібно враховувати.Дані записував через FileStream зі стандартним розміром буфера.

У SQLite був один нюанс на якому вважаю за потрібне акцентувати увагу. Ми не могли складати весь вміст документа в одну клітинку бази даних. Справа в тому, що з метою оптимізації ми зберігаємо текст документа через підрядник, а це значить, що для того щоб скласти текст в одну клітинку нам потрібно було б злити весь текст в один рядок, ніж збільшити, кількість використовуваної оперативної пам'яті. Іншу сторону тієї ж проблеми отримали б і на читанні даних з бази. Тому в SQLite була окрема табличка, де дані зберігалися по рядках та були пов'язані з зовнішнім ключу з таблицею, де зберігалися лише властивості документів. Крім того, вийшло трохи прискорити базу, вставляючи дані блоками по кілька тисяч рядків в режимі синхронізації OFF, без журналирования і в рамках однієї транзакції.

У LiteDB просто віддавався об'єкт, у якого одним з властивостей був List <string> і бібліотека сама це зберігала на диск.

Ще під час розробки тестового додатка я зрозумів, що мені більше подобається LiteDB, справа в тому, що тестовий код для SQLite займав понад 130 рядків, а код вирішальний ту ж задачу для LiteDB менше 40.

Нижче наведені середні результати декількох прогонів тестового коду. При вимірі статистичне відхилення було незначним (Рисунок 3.9).

Після аналізу проведеного тесту між LiteDB над SQLite явним лідером на мою думку стала LiteDB. З отриманих результатів видно що при не великій довжині рядків файли записуються та зчитуються швидше з LiteDB. У випадках середніми і великими файлами на етапі зчитування LiteDB трішки повільніший ніж SQLite, але при записі даних, LiteDB у два рази швидший ніж SQLite.

Test: 1000 files 10-50 lines. Time in ms			
	SQLite	Files	LiteDB
Read	200	1526	193
Write	753	21712	235
Test: 100 files 100-500 lines. Time in ms			
	SQLite	Files	LiteDB
Read	22	398	23
Write	733	3812	233
Test: 10 files 1000-10000 lines. Time in ms			
	SQLite	Files	LiteDB
Read	4	620	15
Write	1759	602	510

Рисунок 3.9 - Тестові результати порівняння LiteDB та SQLite

Тому після аналізу результатів у проекті я вирішив використовувати LiteDB. ЇЇ зручно використовувати з C# і Windows Form та легко інтегрувати у проект. Бібліотека написана на c# і якщо щось було не до кінця зрозуміло або виникли питання про певні методи роботи з нею, то завжди можна було переглянути матеріали та приклади на офіційному сайті. По результатам тестів швидкодії, LiteDB на мою думку теж є кращою.

Висновки до розділу 3

В даному розділі було реалізовано та докладно описано метод пошуку взаємопов'язаних подій, було побудовано граф залежності та пошуку зв'язку між двома подіями за допомогою алгоритму A* та приведено приклад роботи метода для конкретної задачі диплому. В подальшому цей метод буде використовуватися для реалізації програми. В даному розділі було описано засоби які використовувалися для реалізації програми. Опис самої програми буде в наступному розділі.

4 ОПИС РЕАЛІЗОВАНОЇ ПРОГРАМИ

В даному розділі буде описано реалізовану програму для пошуку ланцюгів подій в інформаційній системі. Буде приведений детальний опис функцій та інтерфейсу програми. Також буде представлена діаграма класів програми та покроковий приклад роботи з програмою.

4.1 Збір даних з операційної системи для подальшої обробки

Збір даних з операційної системи є дуже важливим для відновлення подій які відбувалися в операційній системі. Так як якщо правильно зібрати ці дані можна покроково відновити послідовність подій які відбувалися в системі, виявити аномалії, визначити вразливості в системи. Але до збору цих даних треба підійти дуже серйозно, так як неправильний збір цих даних може призвести до их часткової втрати, і через це ми не побачимо повну картину подій яка відбувалася в операційній системі.

4.1.1 Збір не волатильних даних

Non-volatile дані - це те, що залишається незмінним, коли система втрачає потужність або вимикається. Прикладами енергонезалежних даних є електронні листи, документи з обробки текстів, електронні таблиці та різні "видалені" файли. Такі дані, як правило, відновлюються з жорстких дисків [16].

Для моєї дипломної роботи я використовував не волотильні дані для аналізу інформаційної системи. Для зчитування цих даних з жорсткого диску було використав програму Autopsy. За допомогою неї було проведенно:

- пошук за ключовими словами;
- відтворення мультимедіа - зчитано EXIF з фотографій та дивіться відеоролики;

- аналіз реєстру – було визначено останні документи і USB-пристрої;
- відтворення видалених файлів з нерозподіленого простору за допомогою PhotoRec;
- аналіз веб-артефактів - Autopsy призначений для пошуку загальних веб-артефактів у сучасних основних браузерах, зокрема: Firefox, Chrome, Internet Explorer. Autopsy витягує наступну інформацію: Закладки, Cookies, Історія, Завантаження, Пошукові запити. Щоб полегшити пошук даних, результати всіх веб-переглядачів об'єднуються. Отже, якщо ви хочете побачити історію користувача, перейдіть до вузла історії. Вам не потрібно переходити через папки для різних веб-переглядачів, перш ніж ви зможете знайти історію.
- аналіз встановленої операційної системи.

Результати програма виводить на екран з можливість їх перегляду (Рисунок 4.1). Є можливість перегляду подробиць про певну подію, побудувати таймлайн, або вивести діаграму за певний час.

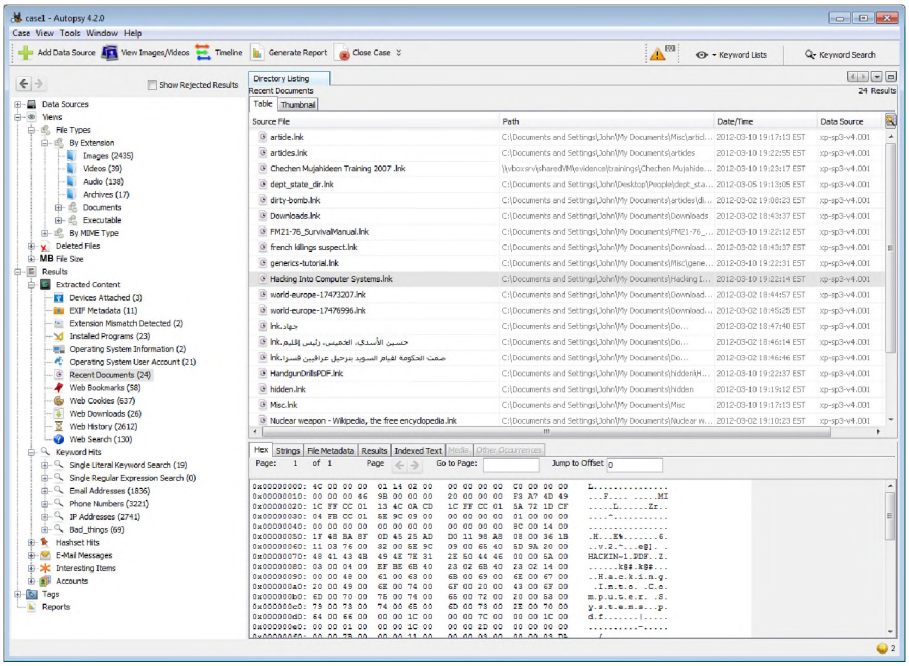

Рисунок 4.1 — Індивідуальний граф взаємозв’язків

Після завантаження даних та їх перегляду є можливість експортувати отримані дані в один із форматів: Excel, HTML, Text, KML, STIX, TSK Body file.

Користувач має можливість вибрати один із цих для експорту, та дані які потрібно експортувати залежно від подальших потреб. Я експортував дані у форматі Excel. Кожний тип даних він зберігає як новий лист, що є дуже зручним для мене для подальшої обробки цих даних.

4.1.2 Збір волатильних даних

Волатильні дані – це дані які будуть втрачені при вимкненні системи. ОЗУ системи містить програми, що працюють у системі (операційні системи, служби, програми тощо) та дані, які використовуються цими програмами. Зміст ОЗУ постійно змінюється і містить багато частин інформації, яка може бути корисною для розслідування. Оскільки оперативна пам'ять та інші волатильні дані є динамічними, колекція цієї інформації має відбуватися в режимі реального часу. Далі приведено нестабільних дані які було отримано з ОС:

- Час системи: аналітики повинні фіксувати час і дату в системі під сумнівом, і це слід порівнювати з фактичним часом і датою. Потрібно відзначити невідповідність. Дати і час запису дозволяють аналітикам документувати, коли починається розслідування інциденту, коли нестабільні дані були зібрані та коли розслідування інциденту закінчилося.
- Мережеві з'єднання: захоплення знімка існуючих мережевих з'єднань дасть аналітикам уявлення про те, які порти відкриті, які процеси встановили з'єднання та підказки про те, які дані було передано.
- Історія команд: перегляд історії команд у підозрілій системі показує недавню активність користувачів і служить слідчим контролем аудиту. Наскільки це можливо, командні історії повинні бути отримані для кожного облікового запису користувача в підозрілій системі. Команди, до яких слід шукати, включають ті, що використовуються для керування обліковими записами користувачів,

конфігурування периферійних пристроїв та встановлення програмного забезпечення.

- Запущені процеси: вивчення списку процесів, що працюють в системі, допоможе аналітикам виявляти зловмисні чи недобросовісні процеси. Під час отримання нестабільних даних аналітики повинні знати, що шкідливі програми можуть мати назви, які вважаються дійсними. Інші ключі, які потрібно шукати, - це процеси, що працюють у непарні часи, і незвичні ідентифікатори користувачів, пов'язані з запущеними процесами.
- Планувальник завдань: є однією з ключових концепцій в багатозадачності і багатопроцесорних систем, як в операційних системах загального призначення, так і в операційних системах реального часу. Планування полягає в призначенні пріоритетів процесам в черзі з пріоритетами. Утиліта, що виконує це завдання, називається планувальником. Найважливішою метою планування завдань є якнайповніше завантаження доступних ресурсів. За допомогою цих даних можна вислідити коли і ким було добавлено завдання, та визначити яку програму чи процес він запускає. Якщо користувач не додавав те чи інше завдання або щось було змінено без його відома це буде вказувати на не санкціонований доступ до ПК.
- Встановлені програми: дає інформацію про повний список встановлених програм, їх версію та місце встановлення. За допомогою цього ми дізнаємося чи встановлювалися за останній період сумнівні програми.
- Інформація про ПК: детальна інформація про ПК, а саме: імя,дата встановлення, ключ системи, терміни дії системи.
- Облікові записи системи: повертає список користувачів операційної системи та інформацію про них: остання активність, права доступу, зміна пароля, статус облікового запису.
- Історія входів користувача в систему: перегляд історії входів в систему, хто і коли запускав цей ПК.

Волатильні дані можна знайти або в Панель управління в ОС Windows або за допомогою команд в CMD або PowerShell чи встановити програмний додаток для зчитування цих даних. Я реалізував у своїй програмі зчитування цих даних за допомогою звертання певної команди до CMD, і повертав відповідь у вигляді рядка string. Я вибрав саме цей варіант зчитування цих даних так як результати легше далі обробляти, та використовувати їх для подальшого аналізу.

4.2 Функції та призначення програми

Реалізована програма є інструментом для комп'ютерної форензіки. Після ознайомлення з програмними продуктами для форензіки які зчитують дані з інформаційної системи та відновлюють послідовність подій, я виявив певні недоліки в їх роботі. Тому я вирішив реалізувати власну програму в якій були виправлені ці не доліки. Програма збирає та аналізує дані з інформаційної системи, будує граф взаємозв'язків між даними, будує часову послідовність між ними, та має багато користувацьких налаштувань які можна змінювати залежно від потреб користувача.

Основні функції програми:

Зчитування волотильних даних з інформаційної системи – реалізовано автоматичне зчитування волотильних даних: запущені процеси, детальна інформація про певний процес, інформація про персональний комп'ютер, Активні з'єднання з мережею, інформація про мережеві адаптери, інформація про облікові записи персонального комп'ютера, детальна інформація про певний обліковий запис, історія входів користувача в систему, заплановані завдання в операційній системі. Зчитування виконується автоматично, користувач лише вибирає які дані потрібно завантажити. Такий підхід є дуже зручним адже волотильні дані зчитуються або запитом в CMD для кожного типу даних окремо, або встановленням додаткового програмного забезпечення для автоматизації цього процесу.

Завантаження не волотильних даних з файлу - не волотильні дані завантажуються з файлу. Збір цих даних є не швидким та можуть збиратися досить довго. На відміну від волотильних даних які потрібно збирати безпосередньо з операційної системи, не волотильні дані можна збирати з жорсткого носія даних навіть якщо операційної системи немає. Для збору цих даних часто використовують більш потужне апаратне забезпечення щоб пришвидчити збір даних, а далі вже аналізують зібрані дані іноді на інших менш потужних пристроях. Не волотильні дані було зібрано за допомогою програмного додатку Autopsy, а результати в подальшому експортовані у файл. В подальшому файл з результатами завантажується у мою розроблену програму.

Вибір конкретних даних які в подальшому будуть оброблятися – в програмі є функція вибори які дані завантажувати а які ні. Залежно від потреб користувач може вибирати які саме дані йому потрібні для обробки. За допомогою цього користувач може зосередити увагу на подіях які уму потрібні для обробки, та прибрати зайві дані. Така функція доступна як при завантаженні волотильних та не волотильних даних, так і при перегляді часової послідовності подій.

Збереження зібраних даних в базу даних – усі зібрані дані сортуються та зберігаються в базу даних LiteDB. Це зручно як для розробника щоб далі використовувати, оброблювати, та відтворювати конкретні дані, так і для користувача адже дані відсортовані за параметрами та пришвидшують їх перегляд та роботу програми в цілому. Основні переваг використання бази даних від збереження даних в оперативній пам’яті комп’ютера яку використовує сама програма під час роботи, це те що оперативна пам'ять не так сильно буде навантажуватися і зменшує шанс перезапису даних які вже були збережені в оперативній паняті. Також використання бази даних буде зручно користувачу, адже йому можна не завантажувати дані при кожному запуску програми, досить зробити це лише при першому запуску.

Обробка отриманих даних з можливість їх перегляду – після завантаження даних користувач має можливість переглянути їх відсортованими за параметрами в окремому вікні програми.

Побудова графу взаємозалежності різнотипних даних – одна з основних функція яка була відсутня в готових програмних додатках. Моя програма будує граф взаємозалежних даних не залежно від їх типу. Вершинами графу є події певного типу даних а ребрами між ними є певний параметр який зв’язує ці дані. За допомогою цього графу можна відстежити пов’язані між собою події.

Побудова графу з явними та не явними взаємозв’язками між даними – під час побудови графу взаємозв’язків, він ображається в два етапи. На першому етапі ображається граф з зв’язками які точно є між даними та з не явними зв’язками, тобто з тими які можуть бути а можуть і ні. На другому етапі зв'язки перевіряються є вони між даними чи ні, якщо зв'язок є то він стає явним, якщо зв'язок між даними не підтвердився то ді ці дані взагалі не пов’язані.

Побудова та перегляд timeline різнотипних даних – в програмі є функція побудови часової послідовності подій не залежно від типів даних. Користувач може відстежити яка подія відбувалася за якою та вибрати дані по яким будувати timeline. Також можна налаштувати часові рамки, тобто вибрати період часу за який відображати послідовність.

Збереження отриманих даних у файл – в програмі є функція збереження даних. Можна зберігати у файл наприклад зібрані волотильні та не волотильні дані щоб в подальшому їх використовувати в іншому програмному додатку наприклад, збереження графу для подальшого відтворення або перегляду. На мою думку це е зручно адже користувачу не потрібно буде аналізувати одні й ті самі дані декілька раз. Також ці дані можна використовувати для формування звіту про виконану роботу.

Відкриття даних з файлу – в програмі є можливість зчитування з файлу не волотильних даних, та можливість відкриття попередньо збереженого проекту створеного в моїй програмі.

Програму розроблено з візуальним інтерфейсом реалізованим на Windows Forms з використанням.NET Framework для зручного використання користувачем.

Детальна інформація про програми, зображення її основних блоків та варіантів налаштувань буде описано нижче у розділі.

4.3 Діаграма класів та їх опис

В цьому підрозділі буде приведено діаграму класів реалізованої програми, та детальний опис кожного з класу.

Діаграма класів (Рисунок 4.2):

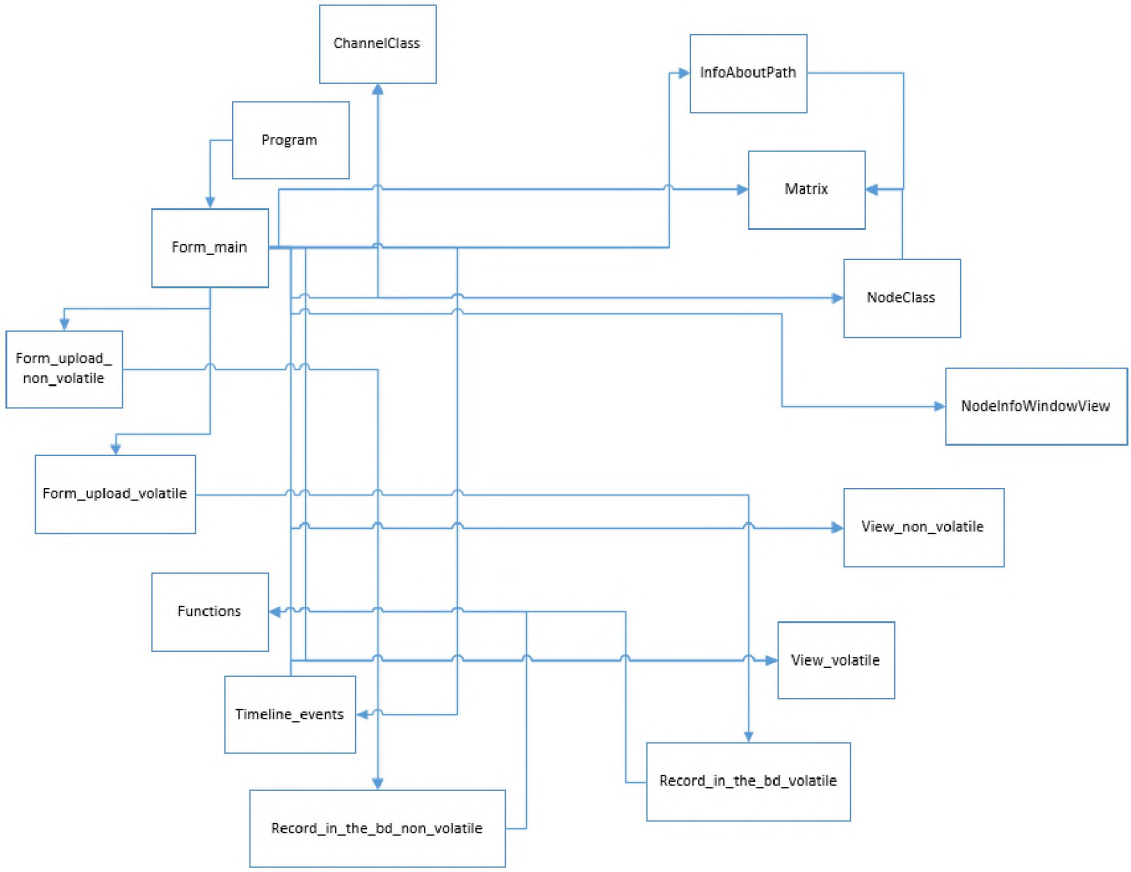

Рисунок 4.2 - Діаграма класів програми

Опис кожного класу:

Program – ініцеалізує компоненти програми;

Form_main – графічне відображення головного вікна програми, виклик інших класів, вимальовування графу взаємозв’язків, відображення викидного меню з іншими можливостями програми;

Form_upload_non_volatile – клас який відображає панель завантаження не волатильних даних та відправляє дані на подальший запис у базу даних, обробляє данні які завантажуються з Autopsy, сортує їх та приводить в однорідний вигляд;

Form_upload_volatile – клас який відображає панель завантаження волатильних даних та відправляє дані на подальший запис у базу даних, обробляє данні які завантажуються з CMD та PowerShell, сортує їх та приводить в однорідний вигляд;

Record_in_the_bd_non_volatile – створює та записує не волатильних даних до бази даних, якщо база вже створена то дописує дані;

Record_in_the_bd_volatile – створює та записує запис волатильних даних до бази даних, якщо база вже створена то дописує дані;

Functions – виконує запити до CMD та PowerShell, та повертає як парамтр string їх відповідь, зчитує з файлу дані з Autopsy;

InfoAboutPath – клас який зберігає інформацію про шлях в графі;

Matrix – який будує матрицю для графу;

NodeClass – зберігає інформацію про всі вершини, їхні параметри, колір та розташування на головному вікні програми;

NodeInfoWindowView – виводить детальну інформацію про кожний вузол

Timeline_events –графічне відображення послідовності подій, будує часову послідовність подій в допоміжному вікні;

View_non_volatile – перегляд завантаженої не волатильної інформації, виводить дані у таблицю у допоміжному вікні;

View_volatile– перегляд завантаженої не волатильної інформації, виводить дані у таблицю у допоміжному вікні;

4.4 Опис інтерфейсу програми

В цьому розділі буде описана програма, її можливості, та описана послідовність дій для використання програми. Після запуску програми буде представлене головне вікно програми (Рисунок 4.3).

В головному блоці можемо побачити доступні до використання наступні опції:

- Головне поле – поле для відображення графу. Вершини графу можна пересувати при необхідності;
- Поля вибору категорії даних та певної події для якої буде будуватися граф;
- Файл – можливість закрити програму, зберігати дані проекту або відкрити проект;
- Дані – відкриває вікно для завантажити волатильні та не волатильні дані, та вікно перегляду волатильні та не волатильні;
- Послідовність подій – відображення послідовності подій даних;
- Очистити – видалення графу та інформації про нього, це потрібно перед побудовою нового графу;
- Додаткове поле в нижньому лівому куті для відображення деталей вибраної події.

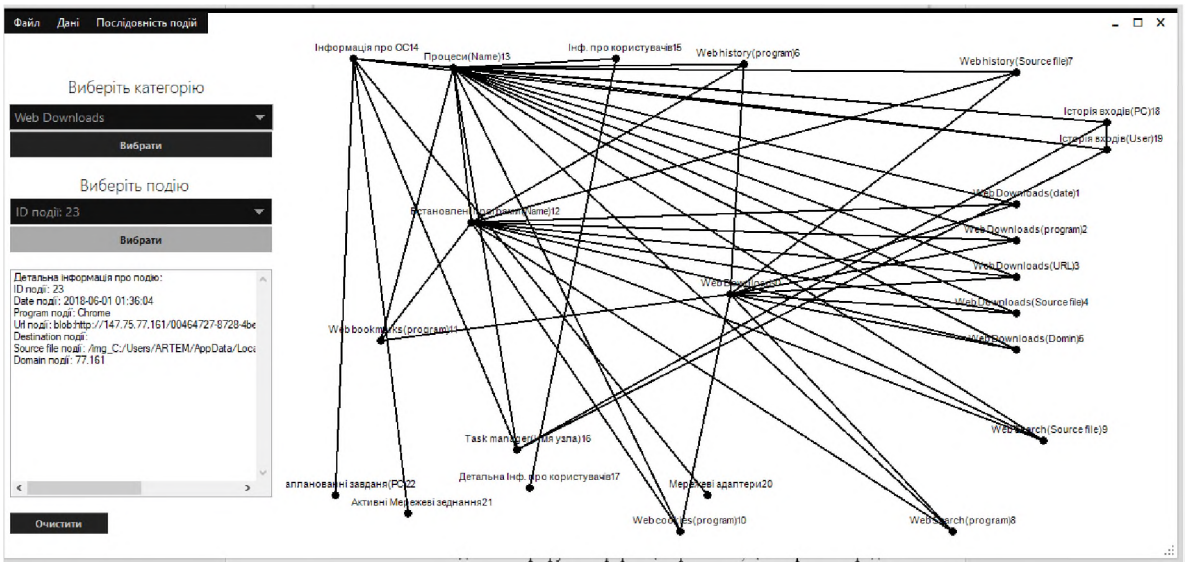

Рисунок 4.3 - Головне вікно реалізованої програми програми

Допоміжний блок для перегляду інформації про вершину графу (Рисунок 4.4). Кожен допоміжний блок відкривається в окремому вікні. Тобто за

допомогою цього користувач одночасно може відкрити їх декілька, наприклад відкрити інформацію про дві чи більше вершини та ознайомитися з результатами або їх порівняти. Якщо натиснути на будь яку вершину то відкриється цей допоміжний блок и покаже детальну інформацію саме про цю вершину:

В допоміжний блок для перегляду інформації про вершину графу можемо побачити доступні до використання наступні опції:

- Поле відображення результату про конкретну вершину. В полі відображається:
- Номер вибраного вузла;
- Тип даних в якому був пошук інформації;
- Параметр по якому був пошук інформації;
- Та список отриманих даних по цій події;

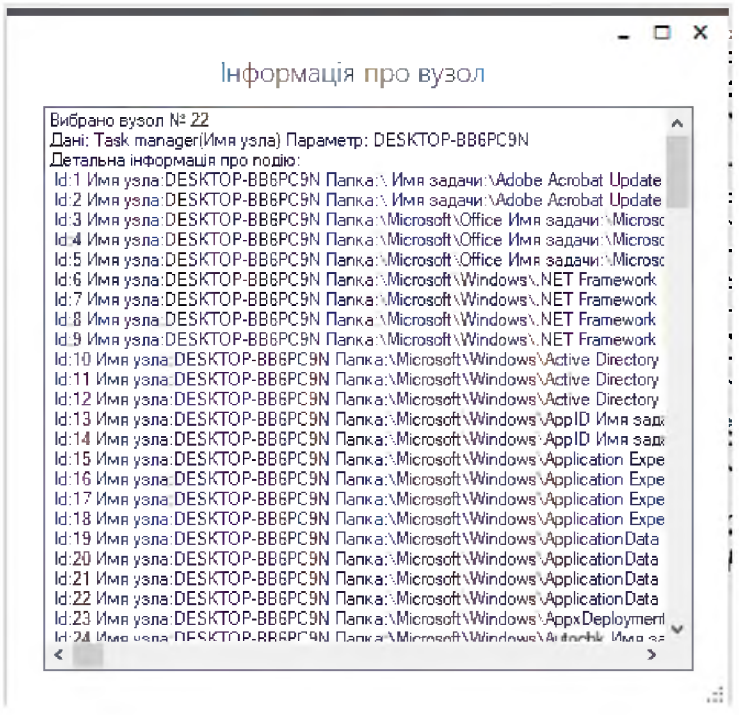

Рисунок 4.4 — Допоміжний блок для перегляду інформації про вершину графу

Блок завантаження волатильних даних та блок завантаження не волатильних даних(Рисунок 4.5):

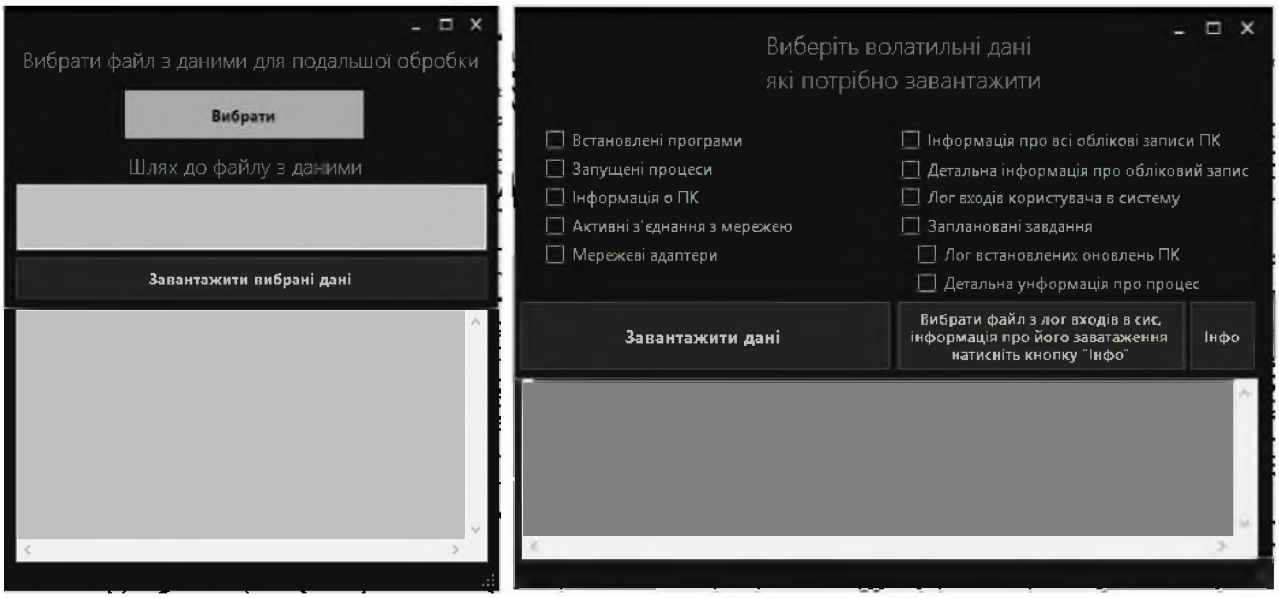

Рисунок 4.5 - Блок завантаження волатильних даних та блок завантаження не волатильних даних

В блоці завантаження волатильних даних можемо побачити доступні до використання наступні опції:

- Панель відображення стану завантаження даних ;
- Панель вибору які дані потрібно завантажити, доступні тауи типи даних для завантаження : встановлені програми, запущені процеси, інформація о ПК, активні з'єднання з мережею, мережеві адаптери, інформація про всі облікові записи ПК, детальна інформація про обліковий запис, лог входів користувача в систему, заплановані завдання, лог встановлених оновлень ПК, детальна інформація про процес;

В блоці завантаження не волатильних даних можемо побачити доступні до використання наступні опції:

- Кнопка вибору файлу з не волатильною інформацією;
- Поле відображення шляху до файлу ;
- Поле стану завантаження даних;

Блок перегляду волатильних даних та блок перегляду не волатильних даних (Рисунок 4.6):

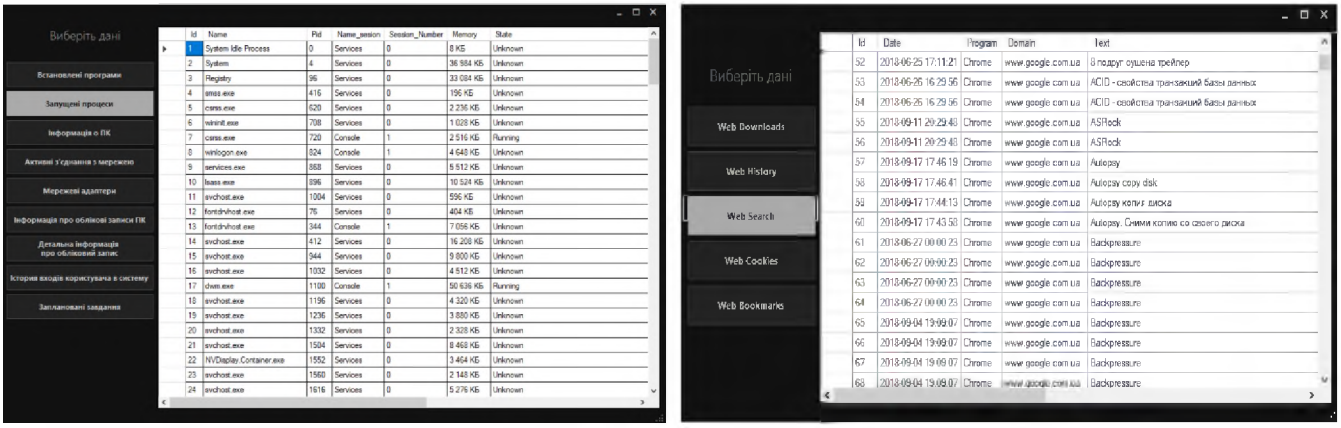

Рисунок 4.6 - Блок перегляду волатильних даних та блок перегляду не волатильних даних

В блоці перегляду волатильних даних можемо побачити доступні до використання наступні опції:

- Вибір для перегляду одного з типів даних які були завантажені. Доступні такі дані для перегляду: встановлені програми, запущені процеси, інформація о ПК, активні з'єднання з мережею, мережеві адаптери, інформація про всі облікові записи ПК, детальна інформація про обліковий запис, лог входів користувача в систему, заплановані завдання, лог встановлених оновлень ПК, детальна інформація про процес;
- Поле відображення вибраних даних у вигляді таблиці;

В блоці перегляду не волатильних даних можемо побачити доступні до використання наступні опції:

- Вибір для перегляду одного з типів даних які були завантажені. Доступні такі дані для перегляду: web downloads, web history, web search, web cookies, web bookmarks;
- Поле відображення вибраних даних у вигляді таблиці;

Наступний блок, це блок відображення послідовності подій (Рисунок 4.7). В цьому блоці відображаються в часовій послідовності вибрані події:

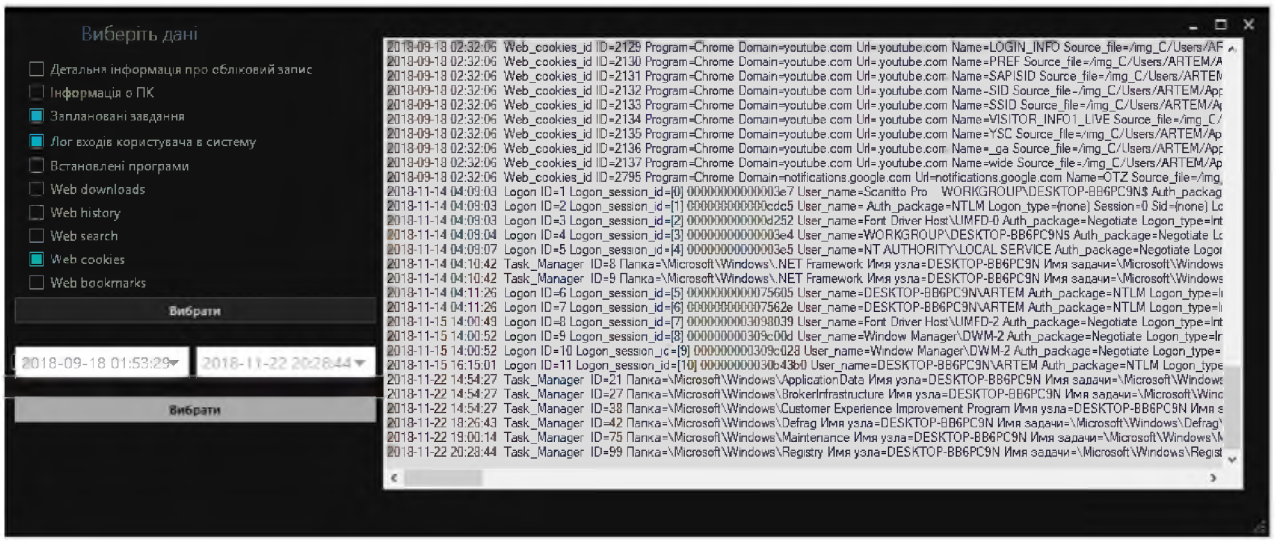

Рисунок 4.7 - Блок відображення послідовності подій

В блоці відображення послідовності подій можемо побачити доступні до використання наступні опції:

- Поле виводу результату відсортованого по часовій шкалі
- Поле вибору даних по яким будувати ланцюг подій. Доступні такі дані для вибору: детальна інформація про обліковий запис, інформація о ПК, заплановані завдання, лог входів користувача в систему, встановлені програми, web downloads, web history, web search, web cookies, web bookmarks;
- Поле вибору часових рамок. Після вибору та завантаження даних стає активним поле вибору з якого по який час відображати події;

4.5 Приклад роботи програми

В цьому підрозділі буде описано приклад роботи з програмою:

1) Завантажити волатильні та не волатильні дані які потрібні для подальшого аналізу.
2) Після цього за потреби можна переглянути завантажені волатильні дані та не волатильні дані, перейдіть у меню (Дані – Перегляд) та виберіть потрібні дані для перегляду.

3) За потреби можна побудувати послідовність подій по часовій шкалі, перейдіть у меню (Послідовність подій) та виберіть дані які потрібно відобразити в часовій послідовності та за який проміжок часу.

4) Після завантаження волатильних та не волатильних даних у систему, користувач може побудувати граф взаємозалежності для певної події. Для цього користувач повинен спочатку вибрати категорію даних в якій знаходиться ця подія.

Після вибору категорії даних користувач може побачити загальний граф взаємозалежності даних, в якому цілісними лініями позначені дані з явним зв`язком, а пунктирними лініями з не явним зв`язком(Рисунок 4.8). За потреби користувач може пересувати вершини графу.

5) Після вибору категорії даних користувач може вибрати саму подію для якої потрібно ввести взаємозалежність. Користувачу будуть доступні всі події в відповідності до категорії яку він вибрав. Події відображені у вигляді Id, який можна переглянути за потреби у блоці перегляду волатильних даних та у блоці перегляду не волатильних даних.

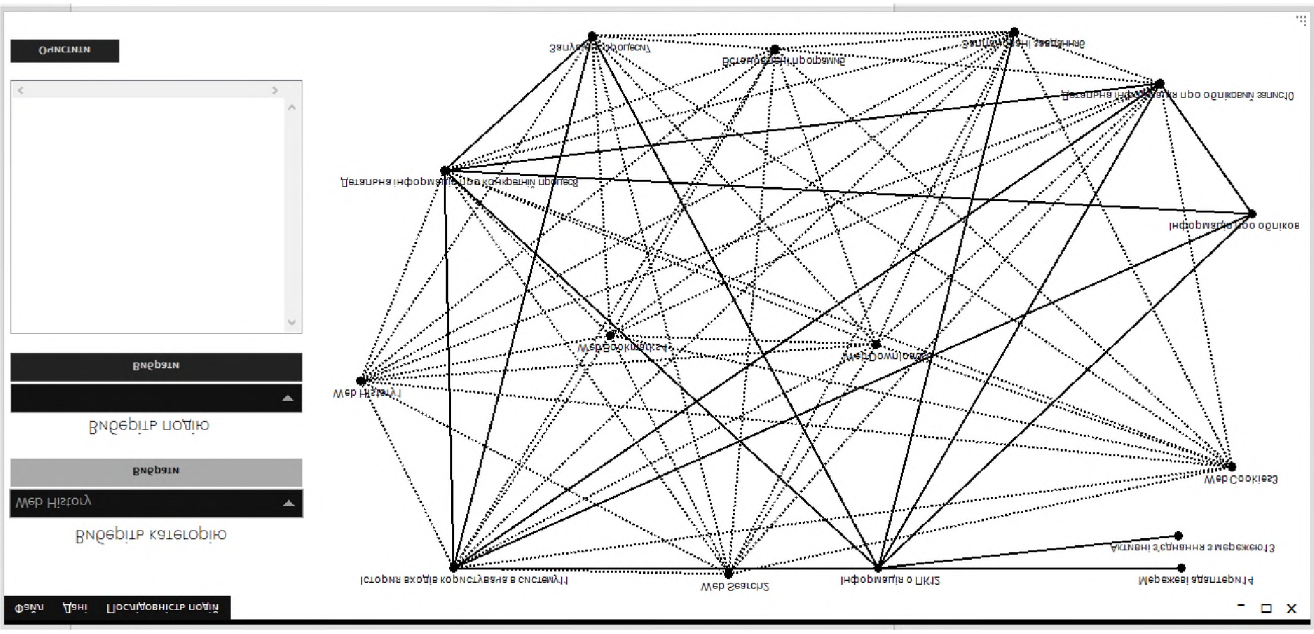

Рисунок 4.8 - Загальний граф взаємозалежності даних в програмі

6) Після вибору конкретного ID події буде побудований граф взаємозалежності для цієї події в головному полі лише з явними взаємозв'язками(Рисунок 4.9).

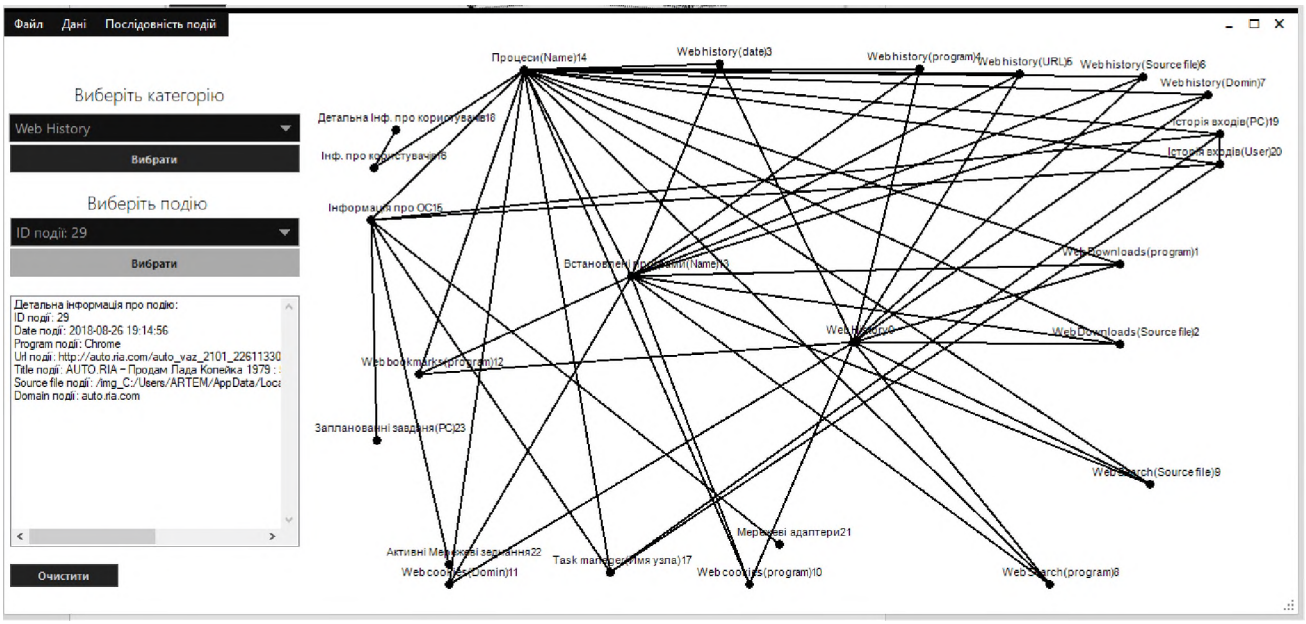

Рисунок 4.9 - Індивідуальний граф взаємозалежності даних в програмі

7) Користувач може переглянути детальну інформацію про подію для якої будується взаємозв'язок у додатковому полі в лівому нижньому куті. Якщо користувач натисне на будь яку вершину то відкриється допоміжний блок який покаже детальну інформацію саме про цю вершину. В разі необхідності користувач може пересувати вершини графу в потрібне йому місце.
8) Якщо потрібно побудувати граф для іншої категорії або для іншої події, потрібно натиснути на кнопку «Очистити», та розпочати дії заново починаючи з пункту 4.

Висновки до розділу 4

В даному розділі було описано реалізовану програму для пошуку ланцюгів подій в інформаційній системі. Програму реалізовано за допомогою метода з розділу 3. Було детально описано функцій та інтерфейс програми. В розділі висвітлено яким чином були отримані волатильні та не волатильні дані для

подальшої обробки. Також в розділі представлена діаграма класів за допомогою якого можна визначити як класи програми взаємопов’язані між собою. В підрозділі 4.5 приведений приклад роботи з програмою, за допомогою якого користувач може зрозуміти послідовність дій під час роботи з програмою.

5 РОЗРОБКА СТАРТАП-ПРОЕКТУ

5.1 Опис ідеї проекту

На першому етапі слід проаналізувати зміст ідеї, можливі напрямки застосування, основні вигоди, що може отримати користувач товару (Таблиця 5.1) і чим відрізняється від існуючих аналогів та замінників (Таблиця 5.2). В останньому випадку - визначення переліку слабких, сильних та нейтральних характеристик та властивостей ідеї потенційної послуги є підґрунтям для формування її конкурентоспроможності.

Таблиця 5.1 - Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Програмний продукт для відновлення ланцюгів подій в інформаційній системі після кібер нападу або збою в системі	1. Захист інформації	Безпека інформації
	2. Зменшення кількості майбутніх атак.	Захищеність інформаційної системи
	3. Розслідування комп'ютерних злочинів	Збір доказів про злочин

Таблиця 5.2 показує, що даний стартап проект є конкурентоспроможним, оскільки переважають сильні сторони у техніко-економічних характеристиках, у порівнянні з характеристиками конкурентних.

Конкурентами є аналогічні програмні продукти. В вже представлених продуктах є низка недоліків які я виправив у своїй програмі. Також більшість готових продуктів є платні або пропонують щомісячну платну підписку.

Таблиця 5.2 - Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів				W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проект	Конкурент1	Конкурент2	Конку-рент3			
1.	Бюджетне фінансування	розробка за рахунок розробника	розробка за рахунок бюджетних коштів	розробка комерційна	розробка за рахунок розробника	відсутність фінансування	часткова бюджетне фінансування	бюджетне фінансування
2.	Використання сучасної техніки	використовується сучасна техніка	використовується застаріла техніка	використовується техніка застарілої конфігурації	використовується сучасна техніка	сучасна комплектація технікою	часткова комплектація технікою	техніка застарілої конфігурації
3.	Належна матеріально-технічна база	розробко проводиться за власні кошти на приватному ПК	бюджетна установа	інформаційний центр	інформаційний центр	інформаційний центр	бюджетна установа	власні кошти на приватному ПК
4.	Підключення до Інтернету	є підключення до Інтернету	є підключення до Інтернету	є підключення до Інтернету	є підключення до Інтернету	без підключення до Інтернету	часткова підключення до Інтернету	є підключення до Інтернету
5.	Налагоджена система реклами продукту	продукт не рекламується	є реклама	продукт не рекламується	є реклама	не реклама	часткова реклама	рекламується
6.	Високий рівень розробки	запропоновані методи та алгоритми є досконалими	розробко не досконала та потребує доробок	запропоновані методи та алгоритми є досконалими	розробко не досконала та потребує доробок	розробко не досконала та потребує доробок	розробко є майже досконалою	запропоновані методи та алгоритми є досконалими

Кінець таблиці 5.2

7.	Професіонали програмісти	розробка проводилася студентом	розробка проводилася групою професіоналів	розробка проводилася професіоналом програмістом	розробка проводилася професіоналом програмістом	розробка проводилася студентом	розробка проводилася професіоналом програмістом	розробка проводилася групою професіоналів
8.	Налагоджена співпраця із бізнес-структурами	ні	проводиться	ні	ні	ні	частково	проводиться

5.2 Технологічний аудит ідеї проекту

Далі необхідно провести аудит технології, за допомогою якої можна реалізувати ідею проекту (технології створення послуги) (Таблиця 5.3).

Таблиця 5.3 - Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Програмний продукт для відновлення ланцюгів подій в інформаційній системі після кібер нападу або збою в системі	Технологія 1 (реалізована програма)	наявні	доступні
2		Технологія 2 (наявність бази досліджень)	наявні	доступні
3		Технологія 3 (база проведення досліджень (випробувань))	потрібно розробити	доступні
4		Технологія 4 (оформлення результатів дослідження)	потрібно розробити	доступні
Обрана технологія реалізації ідеї проекту: є можливою				

5.3 Аналіз ринкових можливостей запуску стартап-проекту

Для врахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій проектів-конкурентів проаналізуємо наявність попиту, обсяг, динаміку розвитку ринку (Таблиця 5.4 - 5.2). За результатами аналізу бачимо, що за попереднім оцінюванням ринок привабливий для входження.

Таблиця 5.4 - Характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	2
2	Загальний обсяг продаж, грн/ум.од	20000
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	не має
5	Специфічні вимоги до стандартизації та сертифікації	не має
6	Середня норма рентабельності в галузі (або по ринку), %	40

На основі проведеного дослідження є можливість стверджувати про привабливість проекту для входження на ринок за попереднім оцінюванням.

Таблиця 5.3 – Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
	Підвищення безпеки інформаційної системи	Слідчі, крупні фірми, центри розвідки, приватні користувачі	Відновлення інформації в інформаційній системі, слідчі – розслідування кібер злоченів, центри розвідки – відновлення подій в системі	Не має

Далі проводиться аналіз ринкового середовища: складаються таблиці факторів, що сприяють ринковому впровадженню проекту, та факторів, що йому перешкоджають (Таблиця №5.4-5.7). Фактори в таблиці подано в порядку зменшення значущості. Також проведено визначення загальних рис конкуренції на ринку (Таблиця 5.8).

Таблиця 5.6 - Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Агресивність конкурентів	вплив на систему	може порушити налагоджену систему розповсюдження
3	Висока вартість продукції	підвищення ціни	підвищить агресивність конкурентів
3	Економічні складності	відсутність фінансування	порушили фінансове забезпечення компанії

Таблиця 5.7 - Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
	Тривале існування	тривале існування на ринку	на ринку дає можливість виходу на нові ринки
	Моніторинг потреб споживачів	розуміючи потреби споживачів, розширювати діапазон продукції, що випускається.	розширення діапазону продукції, що випускається.
	Низька вартість продукції в порівнянні з конкурентами	встановлення низької ціни	пришвидшить вихід на нові ринки
	Трудова міграція фахівців до столиці	Підвищення кваліфікованості спеціалістів	розширення штату

Таблиця 5.8 - Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Вказати тип конкуренції олігополія	На ринку присутні компанії, які зарекомендували себе так, що клієнти націлені на користування лише їх послугами через розрекламованість	Збільшення інформованості потенційних клієнтів про якість послуг, що надаються, та донесення усіх переваг у співпраці з клієнтами-підприємствами
2. За рівнем конкурентної боротьби - національна	Планується розгорнення компанії в межах країни	Готовністю до забезпечення відряджень співробітників для відбору проб
3. За галузевою ознакою - міжгалузева/ внутрішньогалузева	Внутрішньогалузева. Усі методики та технології, що застосовуються у аналізах, є вузькоспеціалізованими та вузько направленими	Необхідно добре зарекомендувати компанію та підвищити авторитет у даній галузі
4. Конкуренція за видами товарів: - товарно-родова - товарно-видова - між бажаннями	Товарно-родова. Конкуренція на рівні технології задоволення потреб. Існує конкуренція з іншими моделями, алгоритмами	методи підвищення оцінювання ефективності контролів безпеки
5. За характером конкурентних переваг - цінова	За умови надання послуг з однаковим переліком пунктів і однаковою якістю, перевага надається більш бюджетним пропозиціям	Поєднання бюджетності та доступності з підтриманням високого рівня якості надання послуг
6. За інтенсивністю - марочна	Марочна.	Запровадження власної марки, яка буде впізнаватися

Більш детальний аналіз умов конкуренції в галузі можна провести за моделлю 5 сил М. Портера (табл. 5.9).

Таблиця 5.9 - Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Навести перелік прямих конкурентів	Визначити бар'єри входження в ринок	Визначити фактори сили постачальників	Визначити фактори сили споживачів	Фактори загроз з боку замінників
Висновки:	На ринку спостерігається тенденція до скорочення кількості підприємств і посилення конкуренції на ринку. Вступ України до СОТ відкрив дорогу іноземним виробникам. Великі компанії з іноземним капіталом постійно збільшують контрольовану ними частку ринку, поглинаючи конкурентів.	Бар'єри входу на ринок є порівняно незначними. Вартість організації бізнесу з виробництва сучасних механізмів відновлення даних в інформаційній системі	Існує чітка залежність від постачальників як якості продукції.	Споживачі мають широку географію і проживають переважно у містах. Покупка програмних додатків та алгоритмів відновлення подій в інформаційній системі часто носить імпульсний характер.	Необхідність донесення до споживачів можливостей програми

З огляду на конкурентну ситуацію, можливість роботи на ринку існує. Щоб бути конкурентоспроможним, проект повинен мати такі сильні сторони, як

доступність, висока якість послуг, що надаються, розповсюдженість інформації про дану марку. У (Таблиця 5.10) визначено та обґрунтовано перелік факторів конкурентоспроможності.

Таблиця 5.10 – Обґрунтування факторів конкурентоспроможності

№	Фактор конкурентоспроможності	Обґрунтування вибору
1	Частка ринку	Враховуючи той факт, що тип родового середовища в галузі – консолідований ринок, тобто існує група компаній, які контролюють разом понад 40% ринку, а також те, що інтенсивність суперництва між діючими конкурентами при низьких темпах зростання ринку є однією з головних сил, які діють на конкуренцію в галузі, одним з найважливіших факторів конкурентоспроможності виступає частка ринку, яку займає виробник. В таких умовах чим більше частка ринку, тим більшими ринковими можливостями володіє виробник.
2	Ціна	Чим вигіднішою є ціна для споживача, тим вірогідніше його вибір.
3	Асортимент	В умовах збільшення інтенсивності між існуючими конкурентами завоювання споживачів відбувається за рахунок нових методів та алгоритмів.
4	Доступ до каналів розподілу	Споживач далеко не завжди проявляє прихильність до певної категорії розробників і дуже схильний до експериментів. В цьому випадку завоювати лояльність споживача дуже складно і ще складніше її утримати. Тому для компаній-виробників ключовими чинниками успіху стає сильна дистрибуція, якісний торговий маркетинг і налагоджена система логістики.
5	Торговий маркетинг	
6	Рівень диференціації ТМ	В умовах ведення конкурентної боротьби на споживчому ринку, де попит є ірраціональним та існує велика кількість виробників і розробників при фактично відсутній різниці між товарами, що пропонуються, ключовим фактором успіху є здатність чітко диференціювати ТМ від ТМ конкурентів, надаючи споживачеві унікальну цінність.

Кінець таблиці 5.10

№	Фактор конкурентоспроможності	Обґрунтування вибору
7	Репутація виробника	Якщо компанія має бездоганну репутацію, особливо у сфері якості своєї продукції, то рівень довіри до неї зростає. Також репутація виробника важлива при виході на ринок з новими товарами, або при виході на нові сегменти, що полегшує позитивне сприйняття новинок.
8	Рівень лояльності до бренду	Чим вище рівень лояльності, тим більше компанія має прихильних, а значить постійних споживачів.
9	Унікальність позиціонування	В умовах монополістичної конкуренції, коли фактор диференціації ТМ є ключовим засобом ведення конкурентної боротьби, важливим є створення та підтримання унікального позиціонування, що створює певний захист від конкурентних зіткнень.
10	Маркетинговий бюджет	Від розміру маркетингового бюджету залежить здатність здійснювати маркетингову стратегію підприємства. Маркетингові заходи мають забезпечувати інші конкурентні переваги такі, як рівень диференціації, лояльності, репутація виробника, дистрибуція та просування в торгових точках.

За визначеними у (Таблиця 5.10) факторами конкурентоспроможності проводимо аналіз сильних та слабких сторін стартап-проекту (Таблиця 5.11).

Таблиця 5.11 - Порівняльний аналіз сильних та слабких сторін «Метод відновлення ланцюгів подій в інформаційній системі»

№	**Фактор конкурентоспроможності**	**Вагові значення фактора (1-20)**	**Рейтинг конкурентів у порівнянні з метод оцінювання ефективності контролів безпеки**						
			-3	**-2**	**-1**	**0**	**1**	**2**	**3**
1	Частка ринку	20		I		III	II		
2	Ціна	15			III		I	II	
3	Асортимент	11	III		I	III			

Кінець таблиці 5.11

4	Доступ до каналів розподілу	10	I		III	II			
5	Торговий маркетинг	15			II	III		I	
6	Рівень диференціації ТМ	10		II	I			III	
7	Репутація виробника	14			III	I	II		
8	Рівень лояльності до бренду	14	II		I		III		
9	Унікальність позиціонування	10			III		II		I
10	Маркетинговий бюджет	14	II	I	III				

Умовні позначки позицій конкурентів:

I - конкурент 1;

II - конкурент 2;

III - конкурент 3.

На основі ринкових загроз та можливостей, та сильних і слабких сторін, виділених у попередній таблиці, складаємо SWOT-аналіз. Це матриці аналізу сильних та слабких сторін, загроз та можливостей (Strength, Weak, Troubles, Opportunities відповідно). Це фінальний етап ринкового аналізу можливостей впровадження проекту (Таблиця 5.12)

Таблиця 5.12 - SWOT-аналіз стартап-проекту

Сильні сторони	**Слабкі сторони**
1. Науково-технічне забезпечення **2.** Перспективи для розвитку **3.** значний рівень диференціації **4.** позитивна репутація виробника; **5.** перспективи збуту продукції **6.** наявність вертикальної інтеграції.	**1.** Конкуренція **2.** Інформованість клієнтів **3.** слабке самозабезпечення фінансовими ресурсами;

Кінець таблиці 5.12

Можливості	Загрози
1. Збільшення попиту 2. Підвищення уваги до безпеки даних 3. Можливість збільшення обсягів реалізації 4. Доступність 5. Розширення штату	1. Загроза працювати без прибутку скорочення платоспроможного попиту 2. Збільшення конкуренції 3. Загроза підвищення цін

Далі розробляємо альтернативи ринкової поведінки (Таблиця 5.13) для виведення стартап-проекту на ринок та орієнтовний оптимальний час ринкової реалізації з огляду на потенційні проекти конкурентів. Альтернативи аналізуються з точки зору строків та ймовірності отримання ресурсів.

Таблиця 5.13 - Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1.	Збільшення фінансування технологічної і інструментальної бази для ще більшого підвищення якості надання послуг програмного продукту	Необхідне залучення стороннього інвестування	Період реалізації може розтягнутися на кілька років
2.	Проведення додаткових рекламних компаній	Можливий пошук безкоштовних рекламних майданчиків або цільове персональне розповсюдження інформації для потенційних клієнтів	Даний варіант потребує 1-6 місяців для досягнення мети

5.4 Розроблення ринкової стратегії проекту

Для розроблення ринкової стратегії в першу чергу необхідним є визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів (Таблиця 5.14).

Таблиця 5.14 - Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Приватні особи	готовий	середній	мінімальна	простий
2	Підприємства які займаються розслідуванням кіберзлочинів	готовий	високий	максимальна	Складність входу у сегмент

Які цільові групи обрано: приватні особи (оскільки є високий попит, а не надто висока конкуренція робить можливим вхід у сегмент) і частково підприємства які займаються розслідуванням кіберзлочинів (через привабливий високий рівень попиту на послугу)

Таблиця 5.15 - Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку*
1	Для кожного сегмента розробити окремі комплексні пропозиції	Стратегія диференційованого маркетинг	Висока якість, бюджетність, доступність	Стратегія диференціації

Таблиця 5.16 - Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки*
	Проект є варіантом вдосконалення відомих підходів, а не принципово новим на ринку	компанія націлена на популяризацію надання послуг серед нових споживачів	частково	наслідування лідеру

Таблиця 5.17 - Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	Висока точність результатів, швидкість визначення, бюджетність	Стратегія диферен-ціації	Стратегія наслідування лідера	Унікальність Якість Бюджетність Реалізація нових методів

4.5 Розроблення маркетингової програми стартап-проекту

Першим кроком є формування маркетингової концепції послуги, яку отримає споживач. Для цього у (Таблиця 5.18) підсумовано результати попереднього аналізу конкурентоспроможності товару.

Таблиця 5.5 - Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1.	Потреба у відновлення даних, виявлення потенційних вразливосте, розслідування кіберзлоченів	Безпека системи	Високий рівень безпеки системи, виявлення вразливосте системи

Таблиця 5.19 - Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові		
I. Товар за задумом	Опис метода відновлення ланцюгів подій в інформаційній системі		
II. Товар у реальному виконанні	Властивості/характеристики	М/ Нм	Вр/Тх /Тл/Е/Ор
	Програмний продукт який готовий до використання		
	Якість: ДСТУ В 7371:2013		
	Пакування – без пакування		
	Марка: Метод оцінювання ефективності контролів безпеки		
III. Товар із підкріпленням	До продажу – рівень розробки		
	Після продажу – Програмний продукт		
За рахунок чого потенційний товар буде захищено від копіювання: захист інтелектуальної власності			

Таблиця 5.6 – Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
	2000-10000 грн	5500	11000	2000-3000

Таблиця 5.7 - Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
	Мінімальна кількість посередників	організовувати широку мережу збуту товару	На рівні всієї країни	Організація системи збуту власними силами без залучення сторонніх посередників через високу специфіку першого етапу (індивідуальний прийом замовлення) та надання послуг

Таблиця 5.82 - Концепція маркетингових комунікацій

п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
	програма відновлення ланцюгів подій в інформаційній системі	інформаційні мережі	Відновлення подій та даних в системі	донести переваги до потенційних користувачів	якості і ціни

Висновки до розділу 5

Після проведення даного маркетингового аналізу можна зробити висновок, що є висока можливість ринкової комерціалізації проекту, оскільки наявний

попит високий, динаміка ринку зростаюча, рентабельність роботи на рику є задовільною. Не зважаючи на існуючі незначні бар'єри входження та стан конкуренції, впровадження з огляду на потенційні групи клієнтів і високу конкурентоспроможність проекту є високоперспективним. Для ринкової реалізації проекту доцільно обрати варіант впровадження варіанту проведення додаткових рекламних компаній за умови можливого пошуку безкоштовних рекламних майданчиків або цільового персонального розповсюдження інформації для потенційних клієнтів. Подальша імплементація проекту є доцільною і являється однією із поставлених задач.

ВИСНОВКИ

У даній дипломній роботі було представлено метод відновлення ланцюгів подій в інформаційній системі за допомогою теорії графів. Було зібрано волатильні та не волатильні дані з інформаційної системи для подальшої обробки, оброблені дані були представлені у вигляді графу подій інформаційної системи. Було проведено теоретичне обстеження існуючих методів пошуку шляхів в графі та вибрано метод A* для пошуку взаємопов'язаних події в графі. Вибрано саме цей метод адже він показав найшвидші результати під час пошуку шляху між вершинами в графі. Також було розроблено програмний продукт, з урахуванням недоліків вже існуючих програм, який виконую такі функції:

- Зчитування волотильних даних з інформаційної системи;
- Завантаження не волотильних даних з файлу;
- Вибір конкретних даних які в подальшому будуть оброблятися;
- Збереження зібраних даних в базу даних;
- Обробка отриманих даних з можливість їх перегляду;
- Побудова графу взаємозалежності різнотипних даних;
- Побудова графу з явними та не явними взаємозв'язками між даними;
- Побудова та перегляд timeline різнотипних даних;
- Збереження отриманих даних у файл;
- Відкриття даних з файлу;

В результаті роботи було отримано програмний продукт який можна впроваджувати для криміналістичного розслідування для відновлення послідовності поді та пошуку взаємопов'язаних даних після кіберзлочену. Він є конкурентоспроможним по відношенню до свої вітчизняними та світовими аналогам, адже в ньому було усунуто недоліки які є в існуючих програмних додатків. Також одним із факторів є те що більшість аналогів є платними що є проблематичним використання їх в компаніях з невеликим бюджетом.

ПЕРЕЛІК ДЖЕРЕЛ

1. Федотов Н.Н Форензика – компьютерная криминалистика [Текст] / Н.Н Федотов.: Москва, 2007 – 360с.
2. Jones R. Internet Forensics [Текст] / R. Jones.: O'Reilly Media, 2005. – 240 с.
3. Diestel R. Graph Theory [Текст] / R. Diestel.: NY: Springer-Verlag, 2005. – 422 с.
4. Харари Ф. Теория графов [Текст] / Ф. Харари — М.: Мир, 2006. — 296 с.
5. Johansen G. Digital Forensics and Incident Response [Текст] / G. Johansen.: Packt Publishing Ltd, 2017 – 302 с.
6. SQLite [Електронний ресурс] / R. Hipp. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/SQLite - 2000p
7. Threats to Information Security [Електронний ресурс] / R. Garg. - Режим доступу до ресурсу: https://www.geeksforgeeks.org/threats-to-information-security/-2015p
8. Autopsy [Електронний ресурс] / B. Carrier. - Режим доступу до ресурсу: https://www.sleuthkit.org/autopsy/ - 2018p
9. Autopsy Timeline Mode [Електронний ресурс] / B. Carrier. - Режим доступу до ресурсу: http://www.sleuthkit.org/autopsy/help/tl.html - 2018p
10. Forensic toolkit [Електронний ресурс] / S. Bruce. - Режим доступу до ресурсу: https://accessdata.com/products-services/forensic-toolkit-ftk - 2017p
11. Digital Forensics Framework – цифровой криминалистический фреймворк [Електронний ресурс] / S. Jacob. - Режим доступу до ресурсу: https://kali.tools/?p=1227- 28.02.2013p
12. A* Search Algorithm [Електронний ресурс] / R. Belwariar. - Режим доступу до ресурсу: https://www.geeksforgeeks.org/a-search-algorithm/- 04.07.2017p
13. Euclidean_distance [Електронний ресурс] / A. Howard. - Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Euclidean_distance - 02.052011p
14. 5 Ways to Generate a List of All Installed Programs in Windows [Електронний ресурс] / L. Kaufman. - Режим доступу до ресурсу: https://www.makeuseof.com/tag/list-installed-programs-windows/ - 30.03.2018p

15. Компьютерная криминалистика [Електронний ресурс] / В. Петровец. - Режим доступу до ресурсу: http://www.spy-soft.net/computer-forensics/#i-7 - 2016р
16. Non-volatile storage [Електронний ресурс] / M. Rouse. - Режим доступу до ресурсу: https://searchstorage.techtarget.com/definition/nonvolatile-storage - 21.09.2005р
17. ACID [Електронний ресурс] / T. Haerder. - Режим доступу до ресурсу: https://en.wikipedia.org/wiki/ACID_(computer_science) - 13.12.2016р

ДОДАТОК А

Приклад програмного коду реалізованої програми

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using LiteDB;
using System.Data.OleDb;
using System.Diagnostics;
using System.IO;
using System.Management.Automation.Runspaces;
using System.Collections.ObjectModel;
using System.Management.Automation;

namespace Analize_DP_V2
{

    class Functions
    {
        static public string RunScriptPowerShell(string scriptText)//
        {
            string result_all = "";
            Runspace runspace = RunspaceFactory.CreateRunspace(); // создание процесса

            runspace.Open(); // открытие процесса

            Pipeline pipeline = runspace.CreatePipeline(); // создание конвейера

            pipeline.Commands.AddScript(scriptText); //добавление сценария

            pipeline.Commands.Add("Out-String"); // эта команда форматирует вывод. Без нее возвращаются реальные объекты.

            Collection<PSObject> results = pipeline.Invoke(); // запуск сценария

            runspace.Close(); // закрыте процесса
              foreach (PSObject obj in results)
              {

                result_all = result_all + obj.ToString();
              }


            return result_all; // возврат значения
        }

        static public string path_to_non_volatile_data_str;//шлях до файлу XLS
        static public string path_to_volatile_data_logon_str;//шлях до файлу TXT
        static public string constr = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=";
        static public string constr1 = ";Extended Properties=\"Excel 12.0 Xml;HDR=NO;\"";
        static public bool data_loaded_to_database = false;//завантажувалися дані чи ні
        static public Dictionary<string, int> List_headline = new Dictionary<string, int>();//заголовки з XLS
        static public void read_xls(Dictionary<string, int> List1, string List_name, string name)// зчитування даних з хлс
        {
            List_headline.Clear();
            int parametr_0 = -1;
            int parametr_1 = -1;
            int parametr_2 = -1;
```

```
int parametr_3 = -1;
int parametr_4 = -1;
int parametr_5 = -1;

string parametr_str_0 = "";
string parametr_str_1 = "";
string parametr_str_2 = "";
string parametr_str_3 = "";
string parametr_str_4 = "";
string parametr_str_5 = "";

using (OleDbConnection conn = new OleDbConnection(path_to_non_volatile_data_str))
{
   conn.Open();
   OleDbCommand command = new OleDbCommand(List_name, conn);
   OleDbDataReader reader = command.ExecuteReader();
   if (reader.HasRows)//считуем с файла
   {
      bool bool_head = true;
      while (reader.Read())
      {
         if (bool_head)//считуем заголовки с каждого листа
         {
            int count = 0;
            while ((reader.FieldCount) > count)
            {
               List_headline.Add(reader[count].ToString(), count);
               count++;
            }
            bool_head = false;

            ///Web Downloads  Web History   Web Search  Web Cookies  Web Bookmarks
            switch (name)
            {
               case "Web Downloads":
                  parametr_0 = List_headline["Date Accessed"];
                  parametr_1 = List_headline["Program"];
                  parametr_2 = List_headline["Domain"];
                  parametr_3 = List_headline["Source URL"];
                  parametr_4 = List_headline["Destination"];
                  parametr_5 = List_headline["Source File"];
                  break;
               case "Web History":
                  parametr_0 = List_headline["Date Accessed"];
                  parametr_1 = List_headline["Program"];
                  parametr_2 = List_headline["Domain"];
                  parametr_3 = List_headline["URL"];
                  parametr_4 = List_headline["Title"];
                  parametr_5 = List_headline["Source File"];
                  break;
               case "Web Search"://
                  parametr_0 = List_headline["Date Accessed"];
                  parametr_1 = List_headline["Program Name"];
                  parametr_2 = List_headline["Domain"];
                  parametr_3 = List_headline["Text"];
                  parametr_4 = List_headline["Source File"];
                  break;
               case "Web Cookies":
                  parametr_0 = List_headline["Date/Time"];
                  parametr_1 = List_headline["Program"];
                  parametr_2 = List_headline["Domain"];
                  parametr_3 = List_headline["URL"];
                  parametr_4 = List_headline["Name"];
```

```
                                parametr_5 = List_headline["Source File"];
                                break;
                            case "Web Bookmarks":
                                parametr_0 = List_headline["Date Created"];
                                parametr_1 = List_headline["Program"];
                                parametr_2 = List_headline["Domain"];
                                parametr_3 = List_headline["URL"];
                                parametr_4 = List_headline["Title"];
                                parametr_5 = List_headline["Source File"];
                                break;
                            default:
                                break;
                        }
                    }
                    else//читаем остальную инфу
                    {
                        parametr_str_0 = reader[parametr_0].ToString();
                        parametr_str_1 = reader[parametr_1].ToString();
                        parametr_str_2 = reader[parametr_2].ToString();
                        parametr_str_3 = reader[parametr_3].ToString();
                        parametr_str_4 = reader[parametr_4].ToString();
                        if (name != "Web Search")
                            parametr_str_5 = reader[parametr_5].ToString();


                        ///Web Downloads  Web History   Web Search  Web Cookies  Web Bookmarks
                        switch (name)
                        {
                            case "Web Downloads":
                                Record_in_the_bd_non_volatile.Record_Web_Downloads(String.Join("",
parametr_str_0.Split('E', 'S', 'T')), parametr_str_1, parametr_str_2, parametr_str_3, parametr_str_4, parametr_str_5, name);
                                break;
                            case "Web History":
                                Record_in_the_bd_non_volatile.Record_Web_History(String.Join("",
parametr_str_0.Split('E', 'S', 'T')), parametr_str_1, parametr_str_2, parametr_str_3, parametr_str_4, parametr_str_5, name);
                                break;
                            case "Web Search":
                                Record_in_the_bd_non_volatile.Record_Web_Search(String.Join("",
parametr_str_0.Split('E', 'S', 'T')), parametr_str_1, parametr_str_2, parametr_str_3, parametr_str_4, name);
                                break;
                            case "Web Cookies":
                                Record_in_the_bd_non_volatile.Record_Web_Cookies(String.Join("",
parametr_str_0.Split('E', 'S', 'T')), parametr_str_1, parametr_str_2, parametr_str_3, parametr_str_4, parametr_str_5, name);
                                break;
                            case "Web Bookmarks":
                                Record_in_the_bd_non_volatile.Record_Web_Bookmarks(String.Join("",
parametr_str_0.Split('E', 'S', 'T')), parametr_str_1, parametr_str_2, parametr_str_3, parametr_str_4, parametr_str_5, name);
                                break;
                            default:
                                break;
                        }
                    }

                }
            }

        }

    }
    static public string Cmd_command_output(string cmd_command)// Рядку з даними з  команди з CMD
    {
        string output = string.Empty;
        string error = string.Empty;
```

```
                ProcessStartInfo processStartInfo = new ProcessStartInfo("cmd", "/c " + cmd_command);
                processStartInfo.RedirectStandardOutput = true;
                processStartInfo.RedirectStandardError = true;
                processStartInfo.WindowStyle = ProcessWindowStyle.Normal;
                processStartInfo.UseShellExecute = false;
                processStartInfo.StandardOutputEncoding = Encoding.GetEncoding(1251);
                Process process = Process.Start(processStartInfo);
                using (StreamReader streamReader = process.StandardOutput)
                   output = streamReader.ReadToEnd();

                string writePath = @"Help.txt";
                using (StreamWriter sw = new StreamWriter(writePath, false, System.Text.Encoding.UTF8))
                   sw.WriteLine(output);


                return File.ReadAllText("Help.txt", Encoding.GetEncoding(866));
               //                                                                                                  return
Encoding.GetEncoding(866).GetString(Encoding.Default.GetBytes(File.ReadAllText("Help.txt")));
             }
          }
       }


using LiteDB;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Analize_DP_V2
{
   public class Web_bookmarks
   {
      public int Id { get; set; }
      public string Date { get; set; }
      public string Program { get; set; }
      public string Domain { get; set; }
      public string Url { get; set; }
      public string Title { get; set; }
      public string Source_file { get; set; }
   }

   public class Web_cookies
   {
      public int Id { get; set; }
      public string Date { get; set; }
      public string Program { get; set; }
      public string Domain { get; set; }
      public string Url { get; set; }
      public string Name { get; set; }
      public string Source_file { get; set; }
   }

   public class Web_search
   {
      public int Id { get; set; }
      public string Date { get; set; }
      public string Program { get; set; }
      public string Domain { get; set; }
      public string Text { get; set; }
      public string Source_file { get; set; }
```

```
    }

    public class Web_history
    {
        public int Id { get; set; }
        public string Date { get; set; }
        public string Program { get; set; }
        public string Domain { get; set; }
        public string Url { get; set; }
        public string Title { get; set; }
        public string Source_file { get; set; }
    }

    public class Web_Downloads
    {
        public int Id { get; set; }
        public string Date { get; set; }
        public string Program { get; set; }
        public string Domain { get; set; }
        public string Url { get; set; }
        public string Destination { get; set; }//save dir
        public string Source_file { get; set; }
    }
    ///Web Downloads  Web History   Web Search  Web Cookies  Web Bookmarks
    class Record_in_the_bd_non_volatile
    {
        static public void Record_Web_Downloads(string p0, string p1, string p2, string p3, string p4, string p5, string
name)//запис Web_Downloads в lite DB
        {
            using (var db = new LiteDatabase(@"MyData.db"))
            {
                var customers = db.GetCollection<Web_Downloads>("Downloads");
                var customer = new Web_Downloads
                {
                    Date = p0,
                    Program = p1,
                    Domain = p2,
                    Url = p3,
                    Destination = p4,
                    Source_file = p5
                };
                // Insert new customer document (Id will be auto-incremented)
                customers.Insert(customer);
            }
        }
        static public void Record_Web_History(string p0, string p1, string p2, string p3, string p4, string p5, string name)
        {
            using (var db = new LiteDatabase(@"MyData.db"))
            {
                // Get customer collection
                var customers = db.GetCollection<Web_history>("History");
                // Create your new customer instance
                var customer = new Web_history
                {
                    Date = p0,
                    Program = p1,
                    Domain = p2,
                    Url = p3,
                    Title = p4,
                    Source_file = p5
                };
                customers.Insert(customer);
            }
```

```
    }//запис Web_History в lite DB
    static public void Record_Web_Search(string p0, string p1, string p2, string p3, string p4, string name)
    {
        using (var db = new LiteDatabase(@"MyData.db"))
        {
            // Get customer collection
            var customers = db.GetCollection<Web_search>("Search");
            // Create your new customer instance
            var customer = new Web_search
            {
                Date = p0,
                Program = p1,
                Domain = p2,
                Text = p3,
                Source_file = p4
            };
            // Insert new customer document (Id will be auto-incremented)
            customers.Insert(customer);
        }
    }//запис Web_Search в lite DB
    static public void Record_Web_Cookies(string p0, string p1, string p2, string p3, string p4, string p5, string name)
    {
        using (var db = new LiteDatabase(@"MyData.db"))
        {
            // Get customer collection
            var customers1 = db.GetCollection<Web_cookies>("Cookies");
            // Create your new customer instance
            var customer1 = new Web_cookies
            {
                Date = p0,
                Program = p1,
                Domain = p2,
                Url = p3,
                Name = p4,
                Source_file = p5
            };

            // Insert new customer document (Id will be auto-incremented)
            customers1.Insert(customer1);
        }
    }//запис Web_Cookies в lite DB
    static public void Record_Web_Bookmarks(string p0, string p1, string p2, string p3, string p4, string p5, string name)
    {
        using (var db = new LiteDatabase(@"MyData.db"))
        {
            var customers = db.GetCollection<Web_bookmarks>("Bookmarks");
            var customer = new Web_bookmarks
            {
                Date = p0,
                Program = p1,
                Domain = p2,
                Url = p3,
                Title = p4,
                Source_file = p5
            };
            customers.Insert(customer);
        }
    }//запис Web_Bookmarks в lite DB


  }
}
```